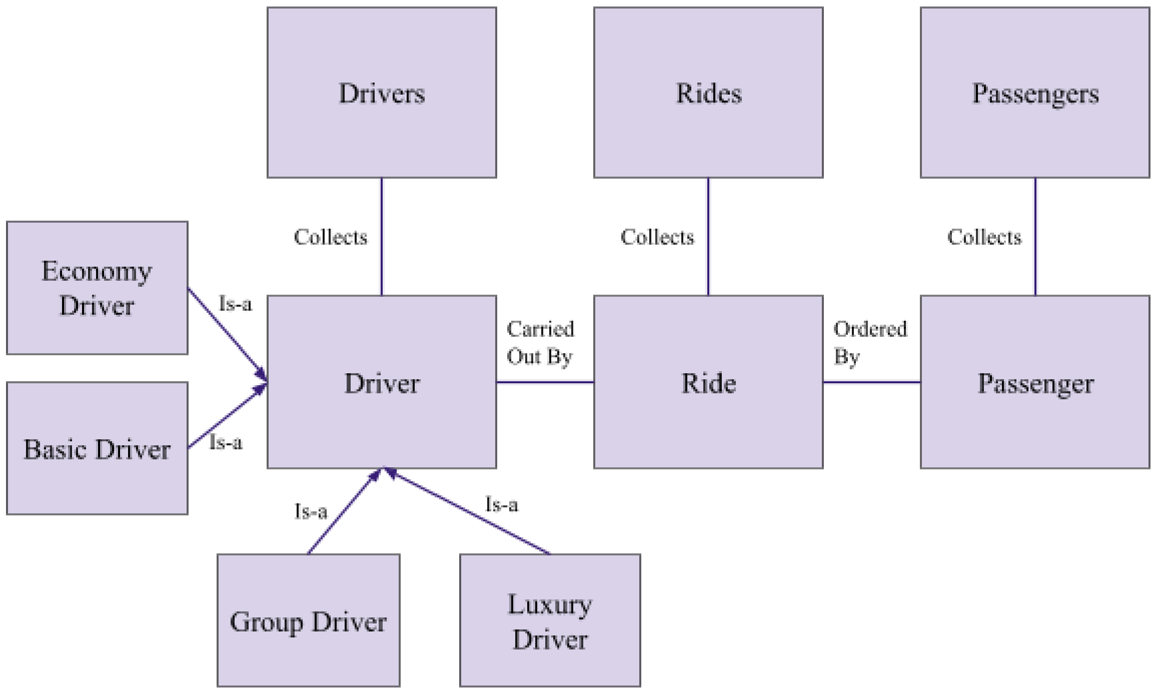
Cathy Du

Homework Four

CSCE 1040.002

C. Du- CSCE 1040 Hwk4 Design and Report

Class Diagram

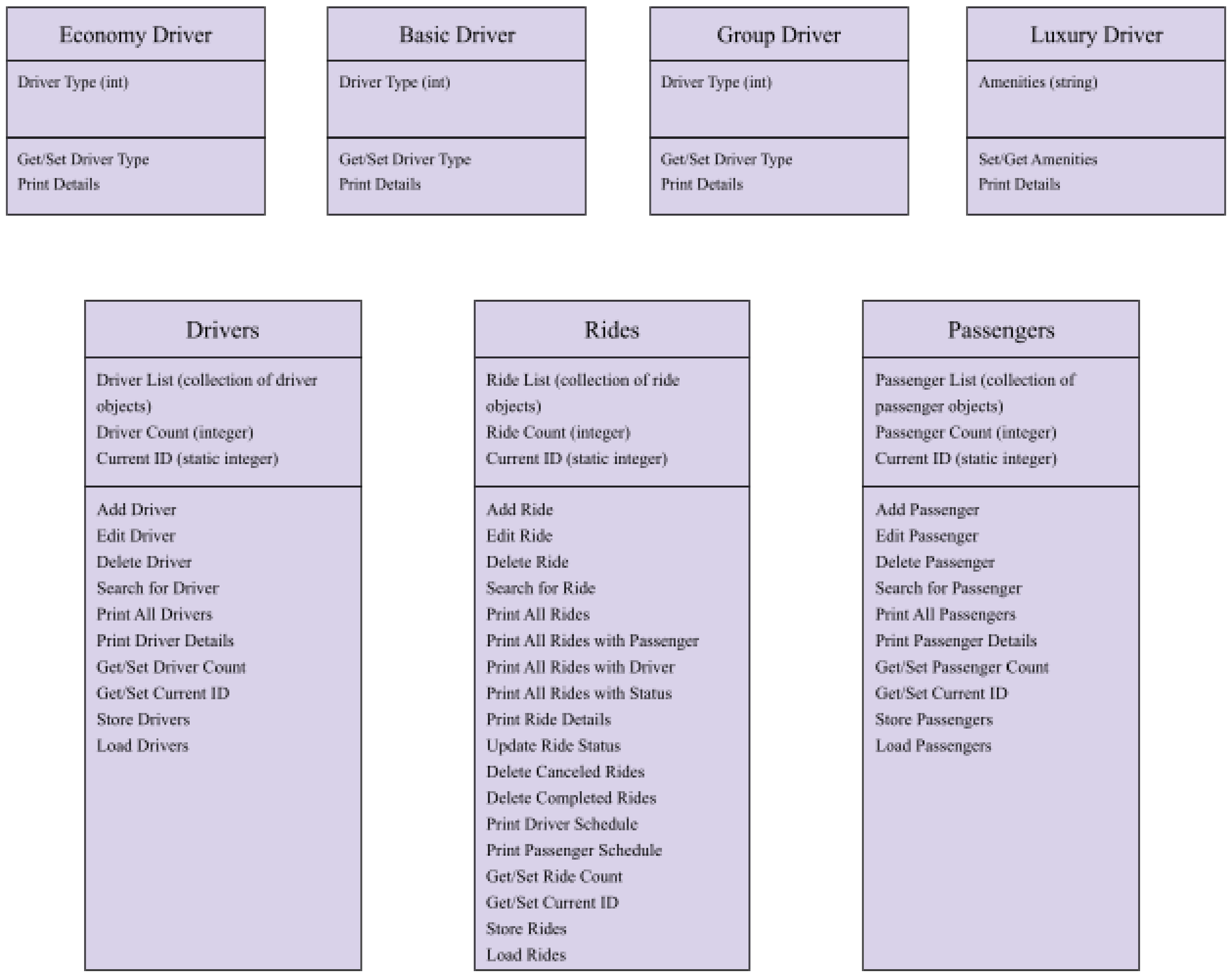


Class Contents

Driver
ID (integer) First Name (string) Last Name (string) Capacity (integer) Cargo Capacity (integer) Handicapped Capable (boolean) Vehicle Type (string) Rating (floating point value) Available (boolean) Pets Allowed (boolean) Notes (string) Type (int)
Set/Get ID Set/Get First Name Set/Get Last Name Set/Get Capacity Set/Get Cargo Capacity Set/Get Handicapped Capable Set/Get Vehicle Type Set/Get Rating Set/Get Available Set/Get Pets Allowed Set/Get Notes Set/Get Type Print Details

Ride
ID (integer) Pickup Location (string) Pickup Time (time value) Drop Off Location (string) Party Size (integer) Includes Pets (boolean) Drop Off Time (time value) Status (string) Rating (floating point value) Passenger (passenger object) Driver (driver object)
Set/Get ID Set/Get Pickup Location Set/Get Pickup Time Set/Get Drop Off Location Set/Get Party Size Set/Get Includes Pets Set/Get Drop Off Time Set/Get Status Set/Get Rating Set/Get Passenger Set/Get Driver Print Details

Passenger
Name (string) ID (integer) Payment Preference (string) Handicapped (boolean) Required Rating (floating point value) Has Pets (boolean)
Set/Get Name Set/Get ID Set/Get Payment Preference Set/Get Handicapped Set/Get Required Rating Set/Get Has Pets operator<< (friend) operator>> (friend) Print Details



Function Pseudocode

Print Details (Driver)

```
Print ID of the driver
Print first name of driver
Print last name of driver
Print capacity of driver
Print cargo capacity of driver
Print handicapped capable for driver
Print vehicle type of driver
Print rating of driver
Print available for driver
Print pets allowed for driver
```

Print notes of driver

<u>Print Details (Economy Driver)</u>

Print Details for driver
Print driver's type (economy)

<u>Print Details (Basic Driver)</u>

Print Details for driver
Print driver's type (basic)

<u>Print Details (Group Driver)</u>

Print Details for driver
Print driver's type (group)

<u>Print Details (Luxury Driver)</u>

Print Details for driver
Print driver's type (luxury)
Print amenities for driver

<u>Print Details (Passenger)</u>

Print name of passenger
Print ID of passenger
Print payment preference of passenger
Print handicapped for passenger
Print required rating of passenger
Print has pets for passenger

<u>operator<< (Friend of Passenger)</u>

(A stream and a passenger as parameters)
Read out id of passenger
Read out name of passenger
Read out payment preference of passenger
Read out handicapped for passenger
Read out required rating of passenger

Read out has pets for passenger
Return the stream

<u>operator>> (Friend of Passenger)</u>

(A stream and a passenger as parameters)
Read in id of passenger
Read in name of passenger
Read in payment preference of passenger
Read in handicapped for passenger
Read in required rating of passenger
Read in has pets for passenger
Return the stream

<u>Print Details (Ride)</u>

Print ID of ride
Print pickup location of ride
Print pickup time of ride
Print drop off location of ride
Print party size of ride
Print includes pets for ride
Print drop off time of ride
Print status of ride
Print rating of ride
Print driver ID of ride
Print passenger ID of ride

<u>Add Driver</u>

Auto assign ID number and increment current ID
Prompt user for driver type
Prompt user for first name
Prompt user for last name
Prompt user for capacity and check that it is valid for the driver type
Prompt user for cargo capacity and check that it is valid for the driver type
Prompt user for handicap capability
Prompt user for vehicle type
Prompt user for rating
Prompt user for availability

```
Prompt user for pet allowance
Prompt user for notes
If the driver type is luxury
    Prompt user for amenities
Create and populate new driver object based on driver type
Add driver object to the driver list
Increment driver count
Store all drivers in driver list in file
```

Edit Driver

```
Prompt user for ID number
Prompt user for data member to update
Check through every driver in driver list
    If driver ID matches
        Check that the value to edit is valid for driver type
        Prompt user for new value
        Update the driver information according to user input
        Store Drivers in file
```

Delete Driver

```
(Driver ID passed as parameter)
Check through every driver in driver list
    If the driver ID matches
        Erase the driver object
        Decrement driver count
        Store all drivers in driver list in file
```

Search for Driver

```
(Driver ID passed as parameter)
Check through every driver in driver list
    If the driver ID matches
        Return the driver object
```

Print All Drivers

```
Check through every driver in driver list
    Print details for driver
```

Print Driver Details

```
(Driver ID passed as parameter)
Search for Driver with the given ID number
Print Details for Driver
```

Store Drivers

```
Open file stream
Check through every driver in driver list
    Write out ID number
    Write out first name
    Write out last name
    Write out capacity
    Write out handicap capability
    Write out cargo capacity
    Write out vehicle type
    Write out rating
    Write out availability
    Write out pet allowance
    Write out notes
    Write out type
    If the driver type is luxury
        Write out amenities
Close file stream
```

Load Drivers

```
Open file stream
While there is input in the file
    Read in ID number
    Read in first name
    Read in last name
    Read in capacity
    Read in cargo capacity
    Read in handicap capability
    Read in vehicle type
    Read in rating
    Read in availability
    Read in pet allowance
    Read in notes
    Read in type
    If driver type is luxury
        Read in amenities
    Create and populate a new driver object based on driver type
    Insert driver object to driver list
    Increment driver count
    Update current ID
Close file stream
```

Add Ride

```
Auto assign ID number and increment current ID
Prompt user for passenger ID
Prompt user for pickup location
Prompt user for pickup time information
Prompt user for dropoff location
Prompt user for party size
Prompt user for pets included
Prompt user for driver ID
Create and populate a new ride object (empty variables for drop off time, status, and rating)
If the passenger exists and the driver exists and the pickup time has not passed and the driver is
available and has enough capacity for the passenger and has an appropriate pet policy for the
passenger and has the passenger's required handicapped capability and has the passenger's
preferred minimum rating
    Add ride object to ride list
    Increment ride count
    Store all rides in ride list in file
```

Edit Ride

```
Prompt user for ID number
Prompt user for data member to update
Check through every ride in ride list
    If ride ID matches
        Check if the data member can be edited
        Prompt user for new value
        Check if the new value is valid
        Update the ride information according to user input
        Store all rides in ride list in file
```

Delete Ride

```
(Ride ID passed as a parameter)
Check through every ride in ride list
    If the ride ID matches
        Erase the ride object
        Decrement ride count
        Store all rides in ride list in file
```

Search for Ride

```
(Ride ID passed as a parameter)
Check through every ride in ride list
    If the ride ID matches
        Return the ride object
```

Print All Rides

```
Check through every ride in ride list
    Print Details for Ride
```

Print All Rides With Passenger

```
(Passenger ID passed as a parameter)
Create a temporary collection of rides
Check through every ride in ride list
    If passenger ID matches
        Add ride to collection
Check through every ride in the temporary collection
    Print Details for ride
```

Print All Rides With Driver

```
(Driver ID passed as a parameter)
Create a temporary collection of rides
Check through every ride in ride list
    If driver ID matches
        Add ride to collection
```

```
Check through every ride in the temporary collection
    Print Details for ride
```

Print All Rides With Status

```
(Status passed as a parameter)
Create a temporary collection of rides
Check through every ride in ride list
    If status matches
        Add ride to collection
Check through every ride in the temporary collection
    Print Details for ride
```

Print Ride Details

```
(Ride ID passed as a parameter)
Search for ride with the given ID number
Print Details for Ride
```

Update Ride Status

```
Check through all rides in ride list
    If the ride drop off time has passed
        Set status of ride to completed
Store all rides in ride list in file
```

Delete Canceled Rides

```
Check through all rides in ride list
    If the ride status is canceled
        Erase the ride object
        Decrement ride count
Store all rides in ride list in file
```

Delete Completed Rides

```
Check through all rides in ride list
    If the ride status is completed
        Erase the ride object
        Decrement ride count
```

Store all rides in ride list in file

Print Driver Schedule

```
(Driver ID passed as a parameter)
Create a temporary collection of rides
Check through each ride in ride list
    If driver ID matches
        Add ride to collection
While the temporary collection is not empty
    Check through every ride in the temporary collection and find first ride
    Print Details for first ride
    Erase the ride object
```

Print Passenger Schedule

```
(Passenger ID passed as a parameter)
Create a temporary collection of rides
Check through each ride in ride list
    If passenger ID matches
        Add ride to collection
While the temporary collection is not empty
    Check through every ride in the temporary collection and find first ride
    Print Details for first ride
    Erase the ride object
```

Store Rides

```
Open file stream
Check through every ride in ride list
    Write out ID number
    Write out pickup location
    Write out pickup time
    Write out drop off location
    Write out party size
    Write out pets included
    Write out dropoff time
    Write out status
    Write out rating
    Write out driver ID
    Write out passenger ID
Close file stream
```

<u>Load Rides</u>

```
Open file stream
While there is input in the file
    Read in ID number
    Read in pickup location
    Read in pickup time
    Read in drop off location
    Read in party size
    Read in pets included
    Read in status
    Read in rating
    Read in passenger ID
    Read in driver ID
    Create and populate a new ride object
    Add ride object to ride list
    Increment ride count
    Update current ID
Close file stream
```

<u>Add Passenger</u>

```
Auto assign ID number and increment current ID
Prompt user for name
Prompt user for payment preference
Prompt user for handicapped status
Prompt user for required rating
Prompt user for pet ownership
Create and populate a new passenger object
Add passenger object to the passenger list
Increment passenger count
Store all passengers in passenger list in file
```

<u>Edit Passenger</u>

```
Prompt user for ID number
Prompt user for data member to update
```

```
Check through every passenger in passenger list
    If passenger ID matches
        Prompt user for new value
        Update the passenger information according to user input
        Store all passengers in passenger list in file
```

<u>Delete Passenger</u>

```
(Passenger ID passed as parameter)
Check through every passenger in passenger list
    If the passenger ID matches
        Erase the passenger object
        Decrement passenger count
        Store all passengers in passenger list in file
```

<u>Search for Passenger</u>

```
(Passenger ID passed as parameter)
Check through every passenger in passenger list
    If the passenger ID matches
        Return the passenger object
```

<u>Print All Passengers</u>

```
Check through every passenger in passenger list
    Print Details for passenger
```

<u>Print Passenger Details</u>

```
(Passenger ID passed as parameter)
Search for Passenger with the given ID number
Print Details for passenger
```

<u>Store Passengers</u>

```
Open file stream
Check through every passenger in passenger list
    Write out passenger
Close file stream
```

Load Passengers

```
Open file stream
While there is input in the file
    Read in passenger
    Add passenger object to passenger list
    Increment passenger count
    Update current ID
Close file stream
```

Report

I started working on this assignment in early November, and have worked on it sporadically over a period of around three weeks. For an exact time, I would estimate around ten hours spent on the assignment, although I am not sure since the time I spent working was so scattered. I found that the homework was not very difficult, since most of the features for the EagleLyft System were already done in Homework 3, and we only had to modify existing code. The modifications we had to make were not as extensive as the original implementations made in the previous homework, and the changes were not particularly complicated or difficult. As a result, this homework was very low difficulty, especially compared to Homework 3.

In terms of problem solving, I learned that problems are usually less intimidating than they seem. One instance that comes to mind is that when I was overloading the insertion and extraction operators, I wrote the functions in the h file instead of the cpp file. I was confused by the error message in the terminal for a long time, but the problem was actually very simple. In terms of time management, I learned that setting larger chunks of time that are farther apart aside for assignments is sometimes more effective than setting aside small chunks of time to work more often. My strategy going into this assignment was to work on it a little bit each day, but I realized that it was hard to get anything done in a short period of time. It was more effective to set aside long periods of time to work on the homework, even if those times were less often. While we will not have another homework for this class, I would like to plan my time accordingly for the next large programming project I do and set aside less often but larger chunks of time to work.

I had the most trouble during this assignment with the open-endedness of the instructions. In my opinion, the previous homeworks had a lot more structure and specific requirements, while this assignment gave a lot of choice as to what to implement in subclasses, which class to overload insertion/extraction operators, and so on. Making my own decisions of how to go about the assignment was probably the part I struggled the most with, since I prefer having clear direction and instructions. Overall, I thought this assignment was not difficult and I am fairly satisfied with what I have done. However, it was notably a lot more open ended than the previous assignments.