Group C – Taxi Company

Link to GitHub Repository - <https://github.com/harrymeighan04/Taxi-Project-Group-C>

16.17

Do you feel the order is appropriate? *No, they are in a random order. You want the simplest features to be first, so that it allows you to incrementally build the application up.*

Is the level of complexity of each too high, too low, or just, right? *I would say most of it is just right, as it’s at an early stage still, and that a lot of the classes have a similar structure, like if its either a taxi or shuttle.* Apart from the final point, as provide all remaining functionality is a bit too open opened to what is required.

Are there any steps missing? *Probably but can’t think of any.*

Revise the list as you see fit, to suit your own view of the project:

■ Enable a single passenger to be picked up and taken to her destination by a single taxi.

■ Enable a single passenger to be picked up and taken to his destination by a single

shuttle.

■ Ensure that details are recorded of passengers for whom there is no free vehicle.

■ Provide sufficient taxis to enable multiple independent passengers to be picked up and

taken to their destinations concurrently.

■ Enable a single shuttle to pick up multiple passengers and carry them concurrently to

their destinations.

■ Ensure that taxis and shuttles are able to operate concurrently.

■ Provide a GUI to display the activities of all active vehicles and passengers within the

simulation.

■ Provide all remaining functionality, including full statistical data.

16.18

Are the completion criteria (tests on completion) for each

stage sufficiently obvious? If so, document some tests for each:

■ Enable a single passenger to be picked up and taken to her destination by a single taxi.

■ Enable a single passenger to be picked up and taken to his destination by a single

shuttle.

* Test to ensure a single passenger can be picked up by a single taxi/shuttle
* Test to ensure a single passenger can be dropped off by a single taxi/shuttle

■ Ensure that details are recorded of passengers for whom there is no free vehicle.

* Test to ensure that there’s a record of each passenger waiting to be picked up

■ Provide sufficient taxis to enable multiple independent passengers to be picked up and

taken to their destinations concurrently.

* Test to ensure more than one passenger can be picked up by a single taxi/shuttle
* Test to ensure than multiple taxi/shuttles can pick up multiple passengers at the same time, and run sententiously

■ Provide a GUI to display the activities of all active vehicles and passengers within the

simulation.

* GUI has been created and all button selections work as expected

■ Provide all remaining functionality, including full statistical data.

* This would be too vague to be tested

16.19

The act method within the Taxi class checks to ensure if it has a valid (i.e., non-null) target location, then moves to the next location. If the next location is the target location, it checks if it currently has a passenger onboard, and if so, offloads the passenger. If the next location is the target location and it does not have a passenger, then it will initiate the process to pick-up the passenger. If it does not have a target, it will increment the idle counter to signify that it has no purpose in life at that current step and will log exactly how many steps it was idle.

16.20

Yes, they should keep seprate lists. They should be kept separate as this would help prevent scheduling issues. You could have a main list that has all active vehicles and another for all occupied vehicles. You would also need a method to move these between lists such as moveTaxiList() in which you move a taxi from idle to active or vice versa.

16.21

I believe the class cannot accept multiple passengers, I think the request pickup method would need to take in a parameter of an array of passengers in which includes the number of passengers, and pass this into the assignments map, in which can be changed to have an array of passengers and then as long as this is a correct value in which can fit in a taxi or shuttle then we can pick up a number of passengers rather than just one.

16.22

Review the way in which *vehicle:passenger* associations are stored in the assignments’ map in **TaxiCompany**. Can you see any weaknesses in this approach? *Yes, only one passenger can be assigned too one vehicle. Also, there’s no way of keeping track to the unique journeys.*

Does it support more than one passenger being picked up from the same location? *No, only one registers passenger can be picked up, but they could 1 or more people with them, but no way of keeping track of them.*

Could a vehicle ever need to have multiple associations recorded for it? *Yes, a shuttle would as it requires the capability to pick up more than one passenger from more than one destination. However, its not required for a taxi.*

Having a map would cause issues for shuttle classes that may have multiple passengers each with individual destinations.

16.23

Including a booking number or identifier would make tracking of each individual passenger to vehicle relationship much easier, making it possible to view each interaction of an instance of a passenger being delivered to an instance of a location, by a specific instance of a vehicle (either shuttle or Taxi). This would also open up reporting as you could record the status of each booking to determine the current number of occupied (i.e., a vehicle that has no active or pending booking) and unoccupied.

16.24

Some tests may still be viable, however if the proposed changes were implemented a number would no longer function without being altered as the structure of the class would be changed so significantly it would require new tests.

16.25

See GitHub for booking number implementation - <https://github.com/harrymeighan04/Taxi-Project-Group-C>