More Parking Garage System Classes for

Master of Science in Information Technology

Software Design and Programming

James McKenna

University of Denver College of Professional Studies

10/19/2025

Faculty: Nirav Shah, M.S.

Director: Cathie Wilson, M.S.

Dean: Michael J. McGuire, MLS

Abstract

This portion of our class assignment is to continue making more classes for our parking system. Incorporating a Money class, Parking Charge class, and a Parking Office class. These new classes needed to be implemented to pull in existing classes such as our Car, Parking Lot, Customer, Address, and potentially Car Type. The new classes were meant to issue Parking Charges that would use the Money class to convert cents into dollars and strings, and finally our Parking Office which would check our parking lots, customers, and charges to validate that our system is tied within itself and working appropriately. All valid unit tests will be attached and a zip file will be included with this assignment to show working code.

CONTENTS

What did you find difficult or easy?

This assignment I found myself having more trouble trying to hook in these new classes into our existing parking system. The Parking Office class incorporated a lot of the other classes into it but also required our new classes to be used in order to build off it. Building new unit tests as well that would test out all the functionality for each of these classes also took a lot of time. I had to develop more unit tests for the setup, registration of new customers, registering of new cars, finding our customers and the customers by their id's, testing chargers for permits, testing charges for invalid permits, testing time steps, testing different amounts, testing all our getters, and also testing the amount and adding the amount correctly to a long and string was actually a lot more work than I thought it would be.

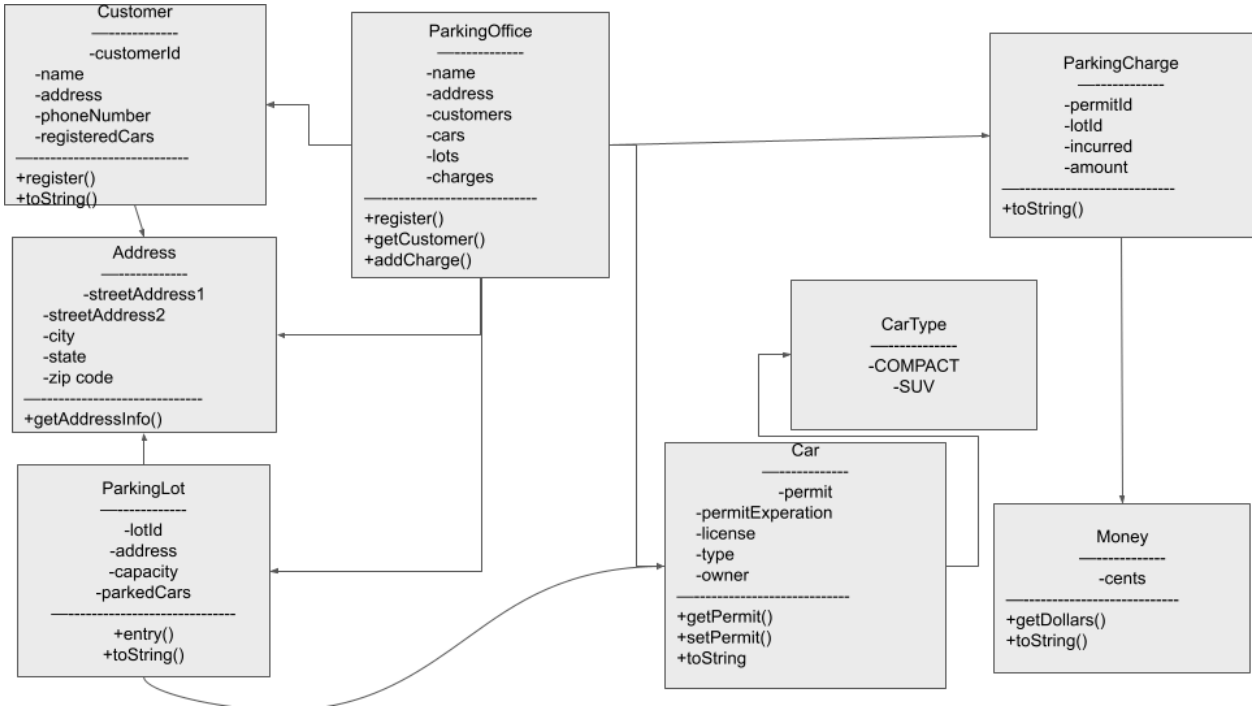
What Helped You?

Personally what helped me the most was the fact that the majority of the classes from the previous assignment set me up for success. I kept it more simplistic and I was very grateful for that. If I added more classes and complexity with this assignment it would have been a lot more arduous for two reasons. The first reason would be building the classes out in such a way that it would be easier to understand and use. The second reason I was having such an issue would be unit testing the complex code. I'm more of a fan of simplistic and easier to understand code for developing purposes. Which is why I'm glad that my previous code was very easy to understand and use on my part.

What did you wish you knew before?

I personally wish in the future that I could make some of my unit tests test more functionality for some of my functions. I ended up writing over 30 unit tests this time around and hope that in the future I can concatenate the unit tests to be able to test more of the classes in an easier fashion. I have a feeling that this class will continue to get harder and harder with new implementations and having to build and test those for our parking garage. Potentially using software design patterns would help lessen the blow on that part.

Implementations and Design Decisions



I had to go back and integrate the new portions of my code into the class diagram. For this I had to clearly show what additions were made over the previous iteration for our class diagram. The biggest addition now is that the Parking Office class is tied into all our major classes for our Customer, Address, ParkingLot and finally our Car classes. This does also tie into our Parking

Charge class but that is more for when and where we have to issue parking charges passed off of incorrect permits or lots. Overall the system, even though it's simple, is starting to build into a more major and complicated system that will require more maintenance and design if this starts to become harder to implement new classes or functionality.

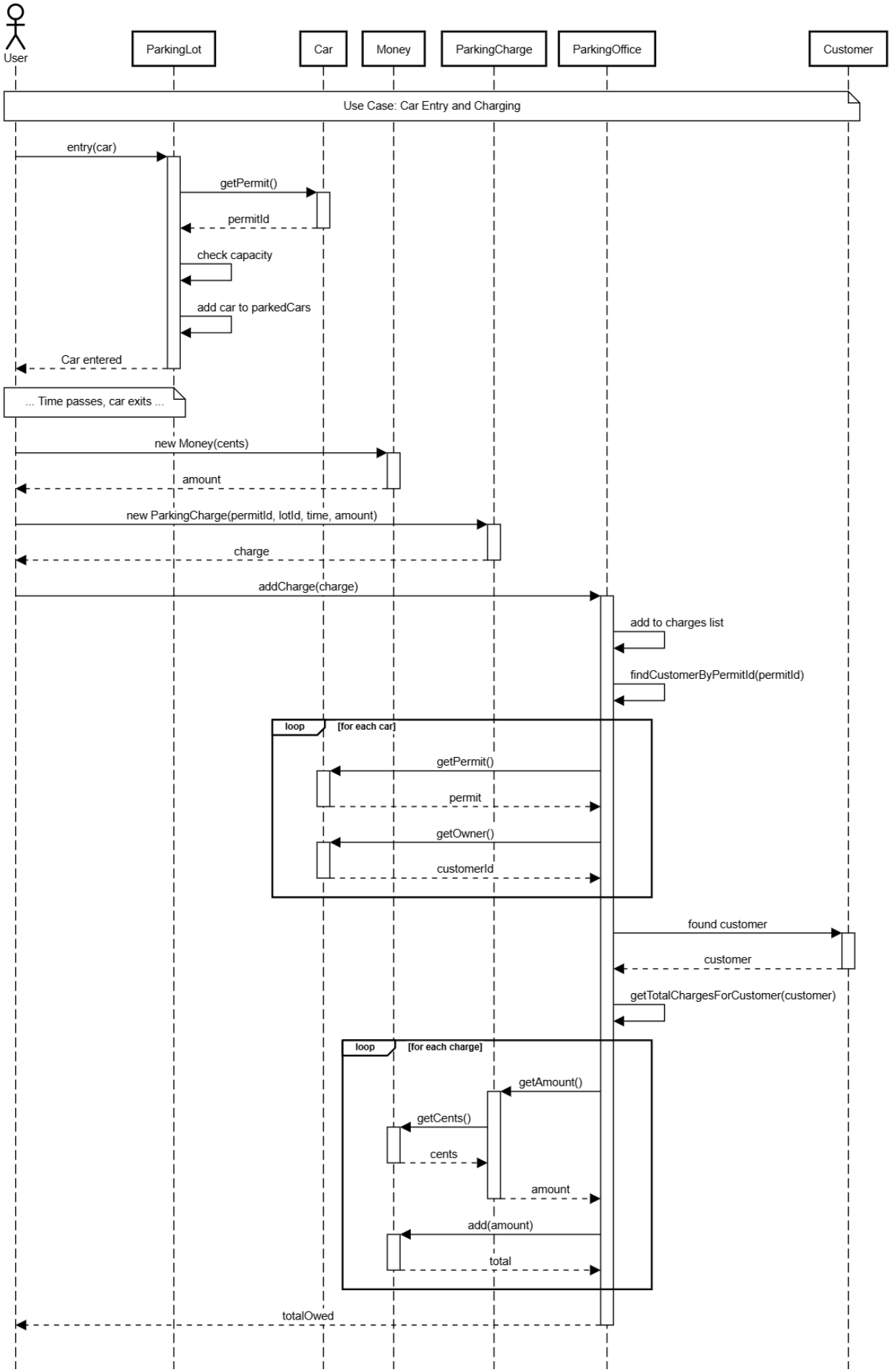

User
ParkingLot
Car
Money
ParkingCharge
ParkingOffice
Customer
Use Case: Car Entry and Charging
entry(car)
getPermit()
permitId
check capacity
add car to parkedCars
Car entered
... Time passes, car exits ...
new Money(cents)
amount
new ParkingCharge(permitId, lotId, time, amount)
charge
addCharge(charge)
add to charges list
findCustomerByPermitId(permitId)
loop
[for each car]
getPermit()
permit
getOwner()
customerId
found customer
customer
getTotalChargesForCustomer(customer)
loop
[for each charge]
getAmount()
getCents()
cents
amount
add(amount)
total
totalOwed

Fig. 1: Charging Guidance Diagram

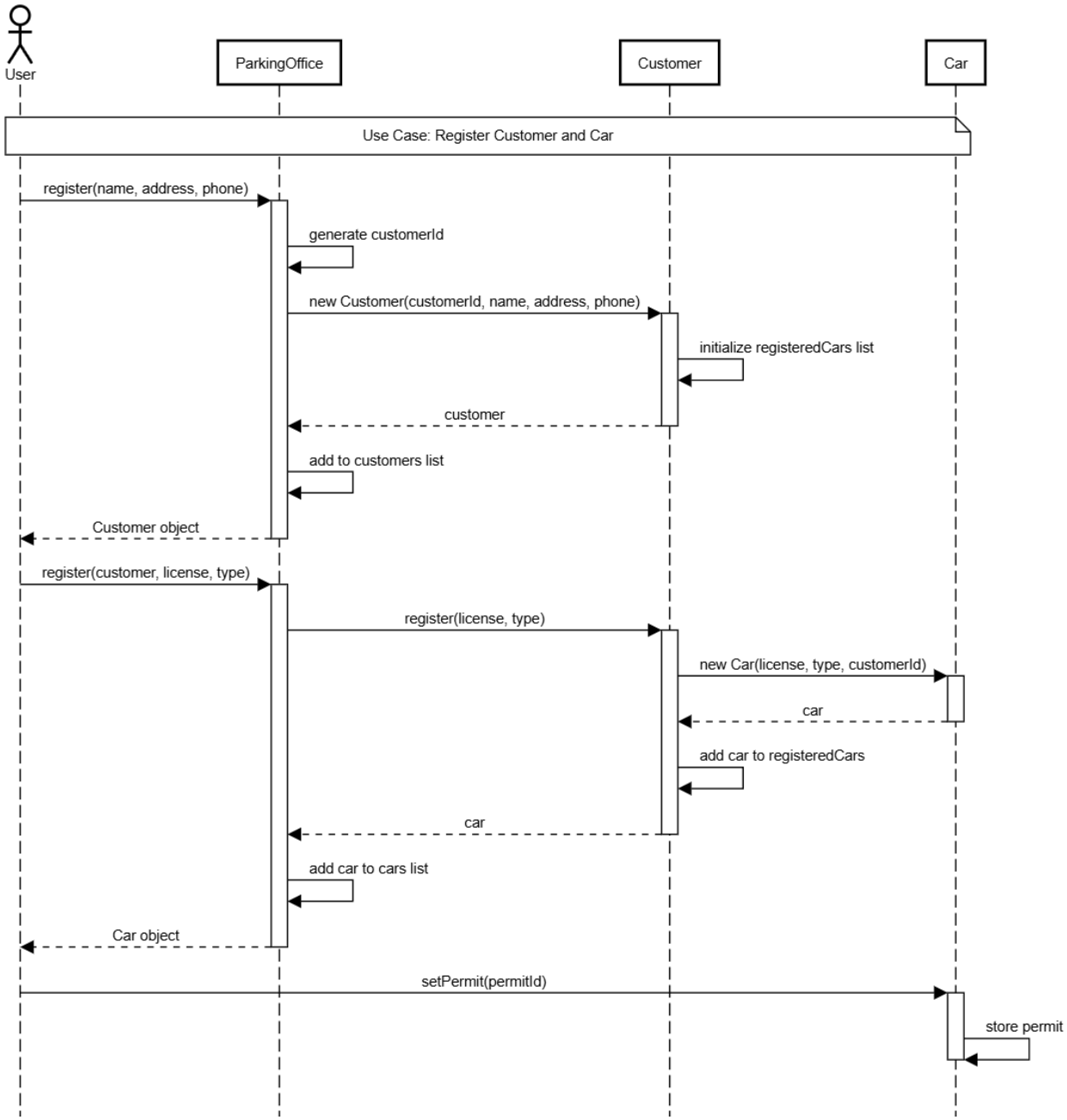


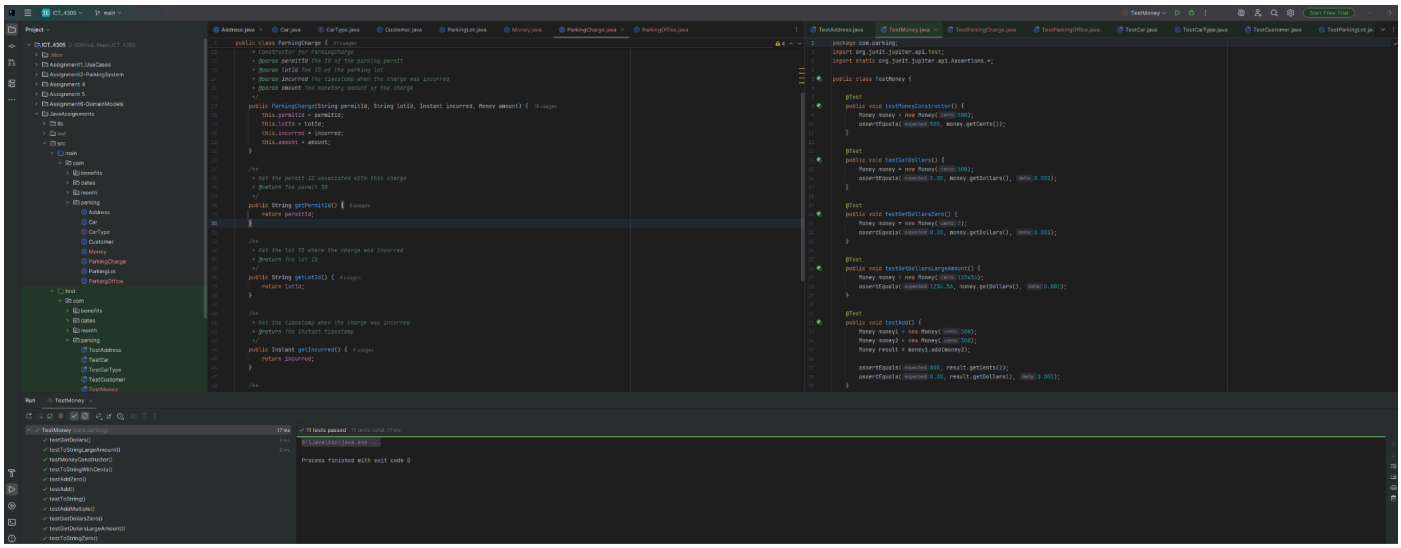
Fig. 2: Registration Sequence Diagram

Lastly, I also included sequence diagrams since that was an additional part of this assignment. I used an online sequence diagram generator to help with the creation of the two sequence

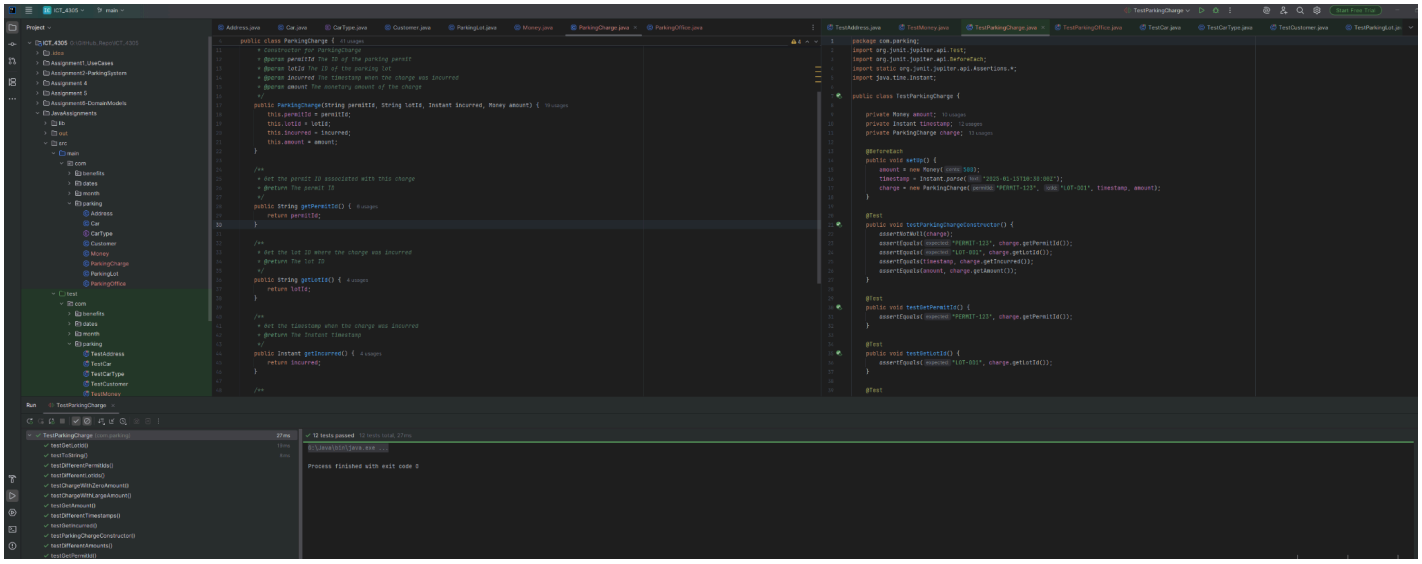
diagrams that show step by step what is happening with our parking garage system. I used one for the registration at our parking system and another one for the charging guidance in our parking system.

Successful Unit Test Screenshots

TestMoney.java



TestParkingCharge.java



TestParkingOffice.java