Week One Assignment

for

Master of Science

Information Technology

Peter Fedor

University of Denver College of Professional Studies

April 6, 2025

Faculty: Nathan Braun, MS

Director: Cathie Wilson, MS

Dean: Michael J. McGuire, MLS

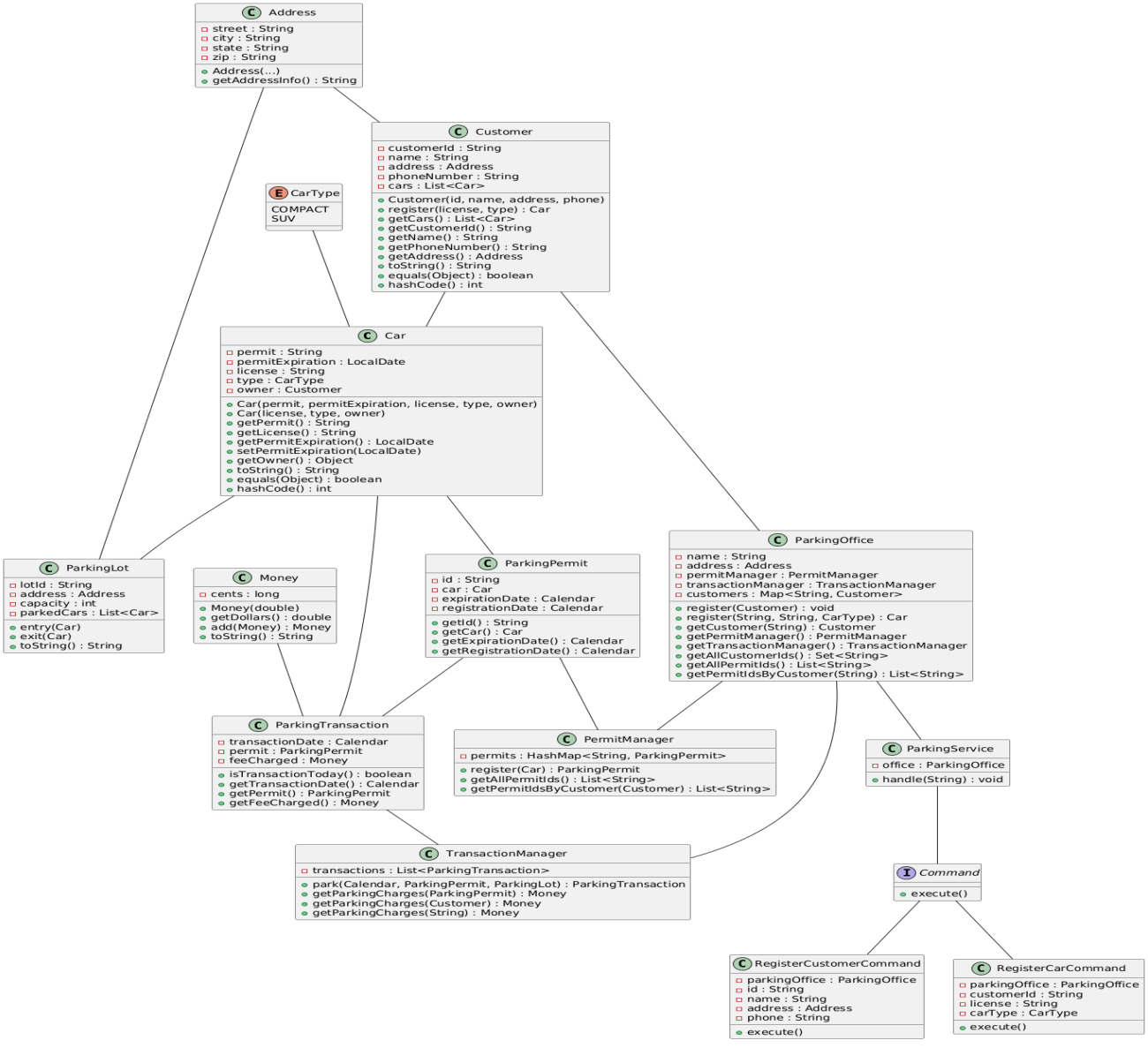
The purpose of this assignment was to expand on the parking system application from ICT 4305 by implementing a command-based user interface. For this, I brought over code from the previous class, Object Oriented Methods, and Programming 1 including classes such as the ParkingOffice, Customer, and Address classes, as well as some Junit tests.

Since the main goal was to add command classes, I will focus on explaining the new components that were created for this purpose. These include the Command interface, the ParkingService class, and two concrete implementations of commands: Register Customer Command and Register Car Command. The command interface is simple, as it is a single method that ensures that all of the commands follow the same structure. This makes the code less coupled because the system will interpret different commands in the same way, making it easier to add more commands in the future if there is a need to scale. It is also loosely coupled because the commands are not hard coded into the program, allowing them to be added and executed independently from the rest of the program. The ParkingService class is as a processing layer for the two command classes I wrote. This is between the user interface and the logical side of the program in the ParkingOffice class. It uses mapping to store available commands and call on them by name, such as CAR and CUSTOMER. They way I view it, is the ParkingService class assists the ParkingOffice class by processing all of the commands. The Register Car Command and Register Customer Command work similarly. First, the RegisterCustomerCommand is responsible for registering new customers. It parses input from the code, such as names and street addresses, creating an address object, and passes that to the parking office. Next, the Register Car Command registers a vehicle to a customer, and issues it a permit for parking. It does so by looking for the customer in the system by name, creates a car object by license plate

number and car type (compact or SUV), then registers the input in the ParkingOffice, returning a permit for the vehicle.

UML Diagram

Here is a UML diagram of my program with class relationships.



I found this assignment to help refresh and expand my Java programming. At first, I struggled a bit with understanding how to implement the command interface, but found good documentation to help me work through the syntax and function of it. Command interfaces are still new to me, and I need to refine my code to work a bit better. I also struggled a bit with GitHub. In ICT-4305 I did not use GitHub to manage my program, so I needed to migrate my entire application to my GitHub repository. There was a good number of resources on the GitHub help site, so this was not as big of an issue. However, I am still not fully comfortable with using GitHub and how it works as a whole.

Since I had to migrate to my GitHub repo, I am still in the process of making sure all of my code is correct, present and behaving as expected. This includes a few Junit tests that I need to move over to my repo. This is to my advantage as I did have to review all of my code again and remember how it all worked together at the expense of a bit of time. Overall, this assignment, though a bit challenging and tedious due to the migration to GitHub, was a great refresher, as well as lessons in both command interface and applying decoupling methods in Java. I am excited to see what else I will learn this quarter.

Bibliography

"Planning Your Migration to GitHub - GitHub Docs." 2025. GitHub Docs. 2025. https://docs.github.com/en/migrations/overview/planning-your-migration-to-github.

w3schools. 2019. "Java Interface." W3schools.com. 2019. https://www.w3schools.com/java/java_interface.asp.