**Parking System Use Cases Reflection**

Software Design & Programming

Object-Oriented Mthd &Pgm I

Joseph DiBiasi

University of Denver College of Professional Studies

10/3/2025

Faculty: Nirav Shah

Director: Cathie Wilson

Dean: Michael J. McGuire

**Introduction**

The university parking system centers around the interaction between the three main classes: car, customer, and parking lot. These classes are a continuation of the previous work designing the use cases for this system. As such, the core concept remains the same. These classes need to let the university register customer cars, scan permits of registered cars, and bill customers monthly for their parking fees.

**Class Design Concepts**

Classes were located in a maven project. Design was based around understanding the default fields and the relationships they implied then expanding each class from that core to meet the university use cases. Cars utilized an owner field which had an implied relationship to the customer id field in the customer class. Both of these were changed from a String to a UUID to better represent a unique ID field. This customer id field is then used in the customer class when a car is registered. Once registered, a car is added to the customers hash map of registered cars. A hash map was chosen because it allows the parking office to easily retrieve all the information needed for a car based on its license field, which will also be unique. This hash map pattern was carried over when the car class was fleshed out.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 1. Compiling Maven Project

The car class contained methods responsible for keeping track of the parking fees incurred by the permit. Locating these methods in the car object itself made it easier to also calculate the monthly bill the parking office sends to customers each month. This was done using a parking fee object which kept track of the fee for each lot individually using the lot id. The parking lot fees were able to be stored in one fashion while the logic was handled elsewhere. The car class could then calculate the total monthly fee using this information and apply the compact car discount if necessary. The customer class was then able to retrieve an aggregate of its total car permit fees instead of worrying about which totals may need a discount.

The parking lot class was the last piece of core logic. Here the main two methods were around entry and exit of cars. Both methods included a lot of validation that would throw errors if conditions were not met. This was an attempt to catch potential hurdles involved with parking such as a customer attempting to use an expired permit or parking in a full lot. Entry and exit parking lots shared some logic common to both but differed in how the daily or hourly rate was charged. A final method was designed to update daily parking lot fees for overnight parking customers. Microsoft copilot was used to generate swift test coverage for classes. These tests were then validated and corrected as needed.

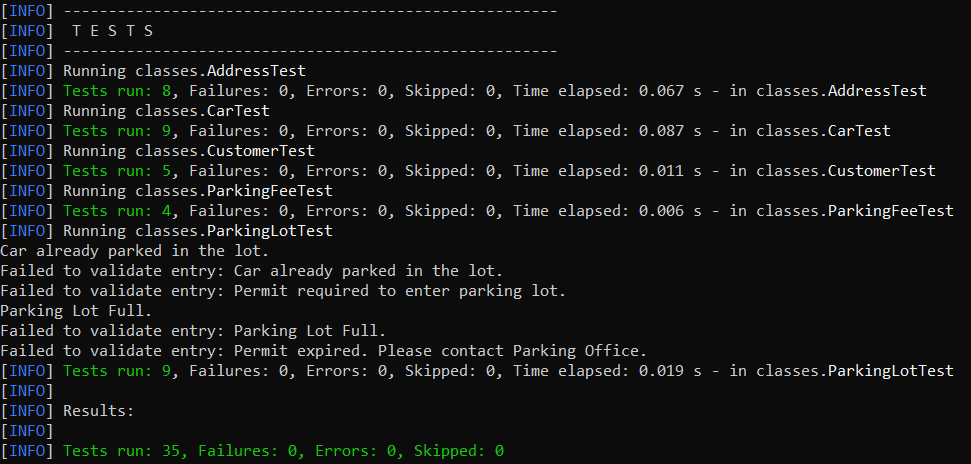


Figure 2. Test Case Results

**Dependencies and Assumptions**

All errors thrown are not caught with the assumption that the greater system will be handling them instead of at the class level. This seemed appropriate as these errors were ones the system should not recover from, such as a customer requiring a permit before attempting to use a parking lot. An additional assumption lies with the license field being unique. License plates should be unique per state, but that is handled by the DMV and not the university. Validating address was not a concern as the customers will be employees and students of the university, their address will be captured and validated by other university systems. Our parking system is an important part of that university, but it is just one part of the greater university network.

**Summary**

Creating the classes for the university parking system was initially difficult as the default fields and methods left several different directions that could be taken to flesh out the classes. However, once the process was started and an initial structure started to take shape development became easier. The initial class design for the parking system gives a solid foundation upon which to build the greater parking system network that would be required by a large university.

**References**

Microsoft Copilot. 2025. “Microsoft Copilot.” Microsoft Copilot. Microsoft. 2025. https://copilot.microsoft.com/.

‌