Implement the Parking Charge Calculator

for

Master of Science

Information Technology

Peter Fedor

University of Denver College of Professional Studies

April 20, 2025

Faculty: Nathan Braun, MS

Director: Cathie Wilson, MS

Dean: Michael J. McGuire, MLS

For this assignment I was tasked with developing a calculator for the parking lot application. The original application assumed there was a flat fee for parking, regardless of the lot. This addition updates it to charge based on the type of vehicle (Compact or SUV), if the vehicle is parked during business hours, which I picked to be 9-5 to keep it simple, whether it is a weekend or not, as well as which type of lot the vehicle is parked in (hourly or daily charge). The reason behind this assignment was to control the parking traffic among the lots and reduce congestion in the lots not being used as much. To achieve the goal of adding these functionalities, I used strategy pattern in order to calculate what a customer would be charged.

Design

For my design of the new calculation additions, I created a new interface called StrategyInterface, which is a single method to define all of the data that would be needed to calculate the charges. This includes the parking lot, permit, and the entry/exit time of the vehicle if applicable. This strategy was then implemented twice in the HourlyCharge and DailyCharge classes. The two different types of lots were handled as they were designed to. For hourly lots, a vehicle is scanned, and the system makes sure the vehicle is not already added to that specific lot, and a time stamp is taken. When the vehicle leaves, it is similarly scanned, and the permit is used to retrieve the entry time to calculate the parking time elapsed. For the daily lots, a vehicle is charged on entrance to the lot. From 12-6am, vehicles are scanned to see if a charge already exists for them. If not, a new charge is added. This is to prevent duplicate charges being added to a vehicle. For the interface to be integrated into the overall application, I updated the parking lot class to reference the StrategyInterface to calculate the corresponding charges. However, I do need to work on the functionality of the existing transaction manager to better handle these

calculations. I struggled to see where these calculations would be stored, so this made it hard for me to conceptualize the entirety of the project, leading us into the next portion of this write up.

Challenges

I found this assignment to be a bit challenging. First off, I found that compared to my prior object-oriented class, there is discrepancy in the code I had written, and what is expected to be used in this class. This made it challenging to decipher what exactly needed to be added to me code for it to work. Next, I have never used Java strategy patterns before and it took a lot of reading to figure out exactly how it works, and how I would implement it into the overall design of the program. This made it hard to figure out where changes in existing classes were needed. I am still trying to map out where I need to add to files within my program. Despite the challenges I faced writing this code, I did find the assignment to be very insightful and helpful for my overall understanding of Java and object-oriented programming.

Conclusion

The new updates to the system allow for future scalability and complexity. The main takeaways I took from this assignment was flexibility and loose coupling. Since the charges are based on an interface, it is easy to add them and increase their capability while decreasing the dependability of hard coded classes to make the calculations. The StrategyInterface can be expanded, allowing the use of new classes to do calculations. I can see this being useful in the future to add additional functionality such as adding an employee, student or guest parking rate. Overall, despite being challenging, this assignment has allowed me to dive deeper into the concepts of object-oriented programming and the capabilities of the Java language.

References

"Design Patterns: Strategy in Java." 2014. Refactoring.guru. 2014. https://refactoring.guru/design-patterns/strategy/java/example.

"Strategy Pattern | Set 1 (Introduction) - GeeksforGeeks." GeeksForGeeks. 2016. GeeksforGeeks. April 29, 2016. https://www.geeksforgeeks.org/strategy-pattern-set-1/.