Using the Observer Pattern to Assess Parking Charges

for

Master of Science

Information Technology

Peter Fedor

University of Denver College of Professional Studies

May 4, 2025

Faculty: Nathan Braun, MS

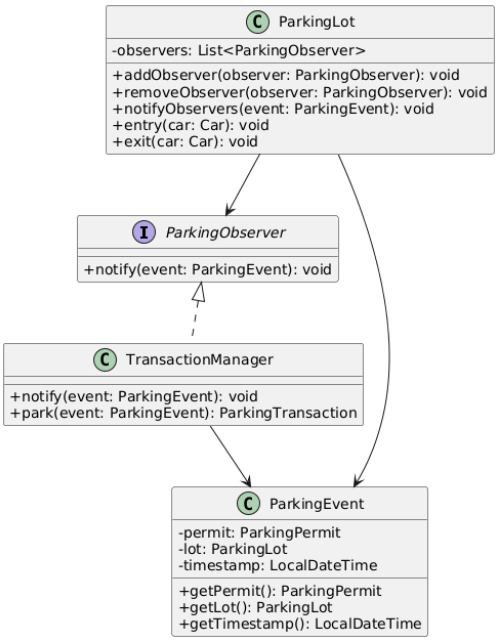
Director: Cathie Wilson, MS

Dean: Michael J. McGuire, MLS

For this assignment I was tasked with implementing an observer pattern into my existing parking lot system. The point of this was to allow the system to respond dynamically to when cars park in the lots on campus. This was done by adding to the parking lot class, as well as the transaction manager, making them the subject and observer respectively. When a car exits the lot, the parking lot then creates a parking event. The parking event collects all relevant details which is then used to create a parking transaction.

Design

For my design, I added a few classes. These being the parking event, parking observer, and parking action classes. The parking observer has a single notify method, which is used when the parking lot class sends a new parking event. The parking event class acts as a sort of template for events, giving observers all of the data points that are needed. The existing parking lot class was modified to include a list of observers with methods that help manage them. The exit method notifies the observers if a car leaves the lot. The transaction manager class was updated to implement the parking observer class, that uses the notify method, while reusing the existing code to log transactions when a charge is received. Junit testing was also added to ensure that the classes were behaving as expected. Here is a class diagram to show the relationship between the new classes and the existing ones.



Challenges

Though observer patterns are new to me, I did not find this assignment to be overly difficult. However, I still did struggle in a few areas. First being the actual implementation of the parking action, event, and observer classes. Since all of these concepts are new to me, I struggled to conceptualize how to actually write out the code. I did need to reference a few resources such as Geeks for Geeks and Baeldung. After a bit of tinkering, I got to all of the classes to interact in the way that I wanted them to. This leads me to another challenge I had, which was updating my existing classes. Admittedly, I struggle with this on a lot of these assignments since I am still not a very confident programmer. I did however find that referencing my class diagram and remembering how my classes interact was extremely helpful.

Conclusion

The observer pattern introduces a new level of functionality to the code. It does so by allowing for the automated reaction of existing code to new events. This decouples and allows for modularity in the entire program. From this assignment, I have realized how the observer

method could be used to further enhance the program, such as automating the analysis of use of each lot. Overall, I think that the use of the observer pattern for our parking lot assignment was a great example of how it can be utilized in other real-world applications.

References

Predrag. 2018. “The Observer Pattern in Java | Baeldung.” Www.baeldung.com. February 11, 2018. https://www.baeldung.com/java-observer-pattern.

GeeksforGeeks. 2016. “Observer Pattern | Set 1 (Introduction).” GeeksforGeeks. April 4, 2016. https://www.geeksforgeeks.org/observer-pattern-set-1-introduction/.

Refactoring Guru. 2014. “Observer.” Refactoring.guru. 2014. https://refactoring.guru/design-patterns/observer.