**Week 6 Assignment - ADD MONEY, PARKING CHARGE, AND PARKING OFFICE CLASSES**

For

ICT-4305 Object-Oriented Method & Program I

Luther Chikumba

University of Denver University College

October 21, 2022

Instructor: Dr. Sherri Maciosek

Table of Contents

[Write-up Summary 3](#_Toc117270186)

[Sequence Diagram 4](#_Toc117270187)

[Class diagrams 4](#_Toc117270188)

[Screenshots 6](#_Toc117270189)

# Write-up Summary

What I discovered challenging to do was debug the code while I was developing the code. I suppose this was because I am used to looking at logs and tracers that help and give quite a lot of information as to why the application system is not working or behaving as it should be. It was a learning curve; I had to challenge myself and adjust the mind frame in which I was working. Debugging the logic in java is quite challenging; finding out where your logic is failing is not straightforward unless you get an error that explicitly tells you what’s wrong with your code. But usually, that is only useful for syntax purposes, much less logic. From that point onwards, I realized that is why developers are encouraged to develop code using the Test-Driven Development (TDD) method, whereby you write your unit tests and then start developing. When I kicked off this week’s assignment, I started developing my code immediately instead of writing tests foremost. I started iterating on the code until I thought I got it right. However, I did not realize that if I get it wrong, I did not know it until I had written the tests, which took a lot of my time cause then I had to go back and scrap almost 3rd of what I had written initially to fix the bugs.

What helped me during this phase was to write many print statements while iterating on my code until I got what I expected or the behavior I hoped to see. Although this helped, I was progressing slowly; debugging many classes in this way was not scalable, and it was inefficient. As these classes are working together, you don’t know if your logic stands as when you change the behavior of one class fixing a bug, you don’t realize that you may have changed the behavior of the other class until you go back and run that class. This is precisely what made learning techniques such as TDD critical in software development.

Before the beginning of this assignment, as this course continues to grow, I wish I had known writing unit tests before diving deep into the actual code. I would have saved time. This is one of the reasons I couldn’t do everything in the assignment that I would have treasured to do: I spent a lot of time trying to debug my code, and yet I was making minimum progress. Whereas if I had used tests to debug my code before delving deep into the code, I would have done a better job instead of depending on unreliable print statements.

Now when it comes to the implementation decisions, the customer had to be registered before registering a car or getting a permit. Now, if the customer is registered only then, the customer can register a vehicle associated with this customer. A registered car can only be associated to one permit; however, the permit can have multiple registered vehicles. Then, the car can be allowed to enter the parking lot.

# Sequence Diagram

Diagram

Description automatically generated

# Class diagrams

|  |
| --- |
| Money |
| cents: long |
| getDollars(): double toString(): String |

|  |
| --- |
| ParkingCharge |
| permitId: String lotId: String incurred: Instant\*\* amount: Money |
| toString(): String |

|  |
| --- |
| ParkingOffice |
| name: String address: String customers: List<Customer> cars: List<Car> lots: List<ParkingLot> charges: List<ParkingCharge> |
| register (String name,             String address,             String phone): Customer register (Customer c,              String license,              CarType t): Car  getCustomer(String name): Customer\*\*\* addCharge(ParkingCharge): Money |

# Screenshots