Report

Analysis

When it came to creating a plan and models for the parking system I wanted to build, my intentions were to create a system that would allow a user (in this case someone who works for the car par for example a car park manager) to create parking meters that would hold a meter id to uniquely identify each meter and it would also hold the parking meter location. The system should allow the user to update/modify meters as well as remove a parking meter. Additionally within the meters that are in the system the user would add parking bands which would be a schedule for times and their prices. The aim is to get the schedule functioning in a way that would allow the meter to update and fetch the correct time/price schedule. The time schedule can be changed or removed if needed and the schedule should hold multiple times and prices, it’s not static as information always changes as do prices which is why it could be modifies or changed if need be.

The second part of the system allows a customer in this instance to enter their duration stay in a parking meter which would again be able to receive the time schedule with the pricing in order to calculate a total charge for the parking ticket. Once it has been displayed and confirmed the system in the meter would also need to validate the payment card in question to authorise the payment and output a parking ticket. This is the plan I came up with when creating my use-case and robustness diagram.

Design

For designing and drawing up the class diagram I references my use case and robustness diagram to help me actually design the system I wanted to build. I had a MeterDAO which is an interface where I had the actual meters coming from like creating/updating/deleting which is used and implemented into many of my other classes and they would use each other. I had a Meter class which uses MeterDAO and the Meter class also uses the Ticket class and each ticket will be referencing a specific meter/ticket machine which would calculate the charge as each meter would be charge different prices. For this reason the Meter class also uses the ParkingBand class as that is where the time schedule (price per half hour) is created and stored so the Meter would use both classes to calculate the charge which is why I created a method in the Meter class called CalculateCharge. The system also has a factory and façade interfaces where the CreateTicket method implements from.

When designing the jsp pages the intention was to design it to allow a user to add Meters so it would display a table that would store and display all the meters (id and location) with buttons to add more meters or modify/delete and existing meters. Also I wanted to create another table for the time schedule which would be displayed underneath the meters and when entering a meterId I would display the specific time/price schedule. With again buttons to allow the user to add new times and prices or edit/remove any existing ones.

Testing

While testing my system, I am able to create meters, update meters and remove them both in my code and jsp pages. My intended strategy for testing was to test as I went along however I ended up doing testing after I had the majority of my code completed and had to fix and test everything all at once. This wasn’t ideal as it created a lot of confusion. My code did not fully match my design as there are gaps which haven’t successfully been tested for example the rest service and my jsp is only completed up to meters and not the specific schedule table I had intended to create. Upon reflection, I would approach the creation of the system completely differently as testing as I went along would have been extremely useful and created less errors to end up with once the coding had been written. Additionally I would approach the coding from a different aspect and logically work through the code instead of doing it all and going back and forth consistently which added to a lot of my confusion. I will definitely be more logical in future.