Report

Decisions made when drawing up domain model and use-case texts

The design section of the project was the most important as this is where I spent most of my time, trying to figure out a way of implementing all the required classes for full functionality. My class diagram went through many iterations, mostly changing where classes linked to data access objects. Originally, I thought to make a class for the ticket that the customer would purchase, however this soon became a problem when trying to implement the code for this along with the schedules and ticket machines.

Decisions made when drawing up sequence diagrams

After creating the class diagram to what I thought was a reasonable diagram of the system, I moved on to make the use case and robustness diagrams. The use case allowed me to map simply map out what each user would be able to do, from creating ticket machines to listing schedules for a specific machine. The robustness diagram was a lot more complex due to how many boundary objects there were, along with all the relevant tests and methods for them. This allowed me to analyse how each class would interact with one another and how the jsps could be used to access these classes.

Detail on places where code did not match design

The original plan was to create the customer interface for purchasing a ticket as a GUI. However I did not have enough time to implement the GUI or even add it as a jsp. Also the ticket schedules did not fully function, particularly when creating it through the add or modify schedule jsp. Originally there were a few variables that were different types, for example the start time for the schedule started as a date but it quickly changed to a string after multiple attempts to get the date type to work properly in my system.

Rationale for test strategy and test plan

A test plan was created for the project to be accessible by anyone, no matter their experience with coding. The test plan consists of step by step instructions on how someone would build the project with maven, how to run the project and mostly how the project functions. If the test plan is carried out correctly, there should be no errors as all the use cases that have been created should be covered in the tests so there shouldn’t be any errors. However if an error occurs, the test plan should explain the code well enough so that the debugging of the code will be smooth.

Critical evaluation of code and/or design.

I believe that the design of the project was done well enough for me to be able to get most of the way through the implementation without many issues. However, I do not believe my implementation of the code was done very well. This is because I ended up not implementing the customer purchasing a ticket, and I struggled to add schedules to a ticket machine. Also, I called the schedules “TicketSchedule” which ended up becoming confusing due to the multiple classes that are very similarly named such as “TicketMachine”, “TicketMachineList” and “TicketScheduleList”.

I decided to change the layout of the original jsp files because it didn’t completely make sense to me. In the end I decided to change the original “ListMachines” jsp to only display a list of ticket machines and allow the creation/deletion of a machine from the list. This made more sense to allow a button to view and then edit the schedule for that specific machine using a separate jsp file.