Design Documentation

Contributors:

Adam Herd - (17364786)
Jaime Kirk - (17922982)

19/03/21

Design Overview

After some initial trouble with how we had set up our threads, we decided to use executors for our threads in this project. We created a thread pool that would allow us to add our tasks to it so that they would run concurrently. In this case, the tasks added were calls to the Mission class, which in turn would create the mission and then run through various functions in the Controller class: move the missions along their stages, send reports and instructions, check for failures along the way etc.

We divided our project into four separate classes: Controller, Mission, Component, and Network. This was suggested in the spec provided by Professor Brennan and we felt it was the clearest and simplest way of tackling this problem. Each class consists of its class variables and getter/setter functions however the controller has a lot more functions as it performs most of the systems features and includes a main function instead of having it in a separate file. We toyed with the idea of having a separate main class instead of having main inside the Controller class but after some testing we deemed it was not entirely necessary so we reverted back to the latter. The number of missions is chosen by the user when the program is run. Therefore, this will determine the number of threads that will run.

For our destinations, we have used the planets in the solar system (as well as the moon) and assigned fuel loads to them based on how far away from earth they are. We then generate each mission with a random fuel load and assign a destination based on a destination map where each possible fuel load is linked with a planet (destination). This system seems to work quite well in practice. Each mission is also assigned an end time. For our network speeds, we divided them by 1000 overall as the program ran very slow otherwise and this allowed us to speed it up somewhat.

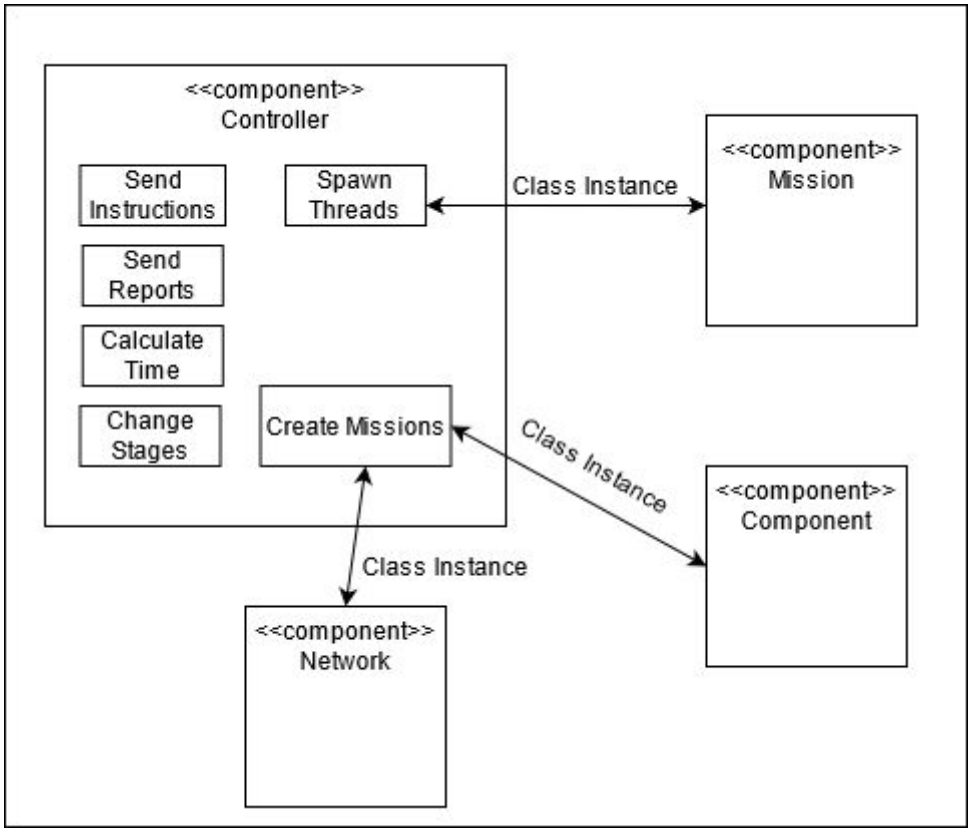
Reports were made up based on what we thought would be relevant information that would be returned in a real system while reports were based on a sample report which was given in the specification.

We've employed locks where our Controller class makes calls to the Network class and Component class as they are shared resources. This was an area we did not fully understand and will need to improve upon going forward. Studying the module notes again did help as well as online resources.

For the division of work, we didn't set out with any detailed plan, just a general overview of the different parts of the system and the knowledge that we would have four classes: Controller, Mission, Component, and Network. We discussed ideas for specific parts and functions of the assignment and would implement them as we went. Naturally, as we implemented more, other parts of the code would break and we would spend time fixing that. Once we had a good start, we formed a list of the remaining tasks to be done and would run through them one at a time until we were confident we had covered everything. This kept the workload fairly even between the two of us.

Component Diagram

Below is a very high level component diagram of our project.



Compile and Run

The programme is quite straightforward to use. The code is compiled using “javac Controller.java” or whatever class you wish to compile. Then to run the system you type “java Controller.java”, this will start the programme and you will enter a number of the amount of missions you wish to run. Press enter after that and the system will start and terminate itself once all missions have run. The data from the programme will be outputted to the “output.dat” file in the directory.

What did we learn: Adam

My main takeaway from this project was the difficulty in running a multithreaded system. I learned better ways to manage threads than I initially thought. Various ways were covered in the course notes and I got familiar with the way I like best which is what we used in the project, that being the use of executors.

My contributions were mixed a lot with my partners. We initially had a thread setup that was not concurrent so I spent some time changing that over to use executors and hence make the threads run concurrently. This involved adding new code and a significant amount of refactoring to get the methods we had already implemented working with the new thread system. It involved a re-read of our course notes as well as some research online for me to fully understand what we needed to do.

Next time I would take great care in setting up the threads first as the refactoring required once we had methods implemented took a great amount of time that could have been saved if we had taken more care in the beginning.

I also worked on the report and instruction sending between the mission components and the controller. This was a simple enough concept to grasp and just involved a few functions and lists to get working. The component list was based on that given in the specification and the instrument list was simply based on what I found on Google about various instruments in space crafts.

From time to time, features we each implemented would break something the other person had done so we commonly fixed each other’s code to ensure everything worked together correctly.

I also need to learn more about locks as I don’t feel I have a good enough grasp on them which may reflect in the code itself.

For the future however, I feel my work on this assignment will be a great foundation for working with the concurrent systems. I have a better grasp of how they work now and would better be able to work with them going forward. There are still some gaps in my knowledge--mainly locks as I've mentioned above--but I feel with a little more practice I can close that gap.

Overall, this project was a nice and fresh challenge as we had not done much work with concurrent systems in the past, especially not on a scale like this. It exposed my lack of understanding of some important areas of software development and it's always enjoyable figuring these things out, no matter how long they might take or frustrate me at the time.

What did we learn: Jaime

The main lesson I learnt from this project was fully understanding how threads and locks work as we have had very little experience using these in previous modules. I had only previously worked with threads once previously so it took a while to get familiar with using them again. I felt that as I progressed in the project my understanding improved and made it easier to understand what was going on in our code.

Studying the module notes helped us understand locks more (though there was still uncertainty) and implement it into our project. Originally we used runnable to implement our threads but then we realised that this way didn't run the threads concurrently so had to change this to executors.

We quite evenly split the work with taking certain functions that we would work on. We initially set up a starting point of the 4 classes with the variables stated in the specification then I worked on getting basic checks such as the 10% failure for mission stages and 25% recovery. This included setting up network features such as availability and network speeds so availability checks could be added as well as calculating sleep time for each mission. This was all done without threads being implemented as we wanted to get basic functionality working first.

After threads were implemented I worked on the tracking of start and end time of each mission, ensuring this was all tracked accurately and the implementation of software updates. This is where I encountered that the start time wasn't being tracked correctly and kept returning 0 for all missions and would never change however after troubleshooting we found this to be a similar issue to what we had with mission destination and just needed to move where the time was being set in the createMissions() function.

For a future project working with threads, I feel I would have a greater understanding of how a concurrent system works so this should reduce the time we spent with issues that we had due to our lack of experience which could have been spent adding extra features that could have improved the system further. A lot of our time was spent changing code we originally used for threads so I feel in a future project we could avoid this and use that time more effectively.