# Refactoring the Supermarket and Bank Queues

For my refactoring I will refactor the simulator. When I was writing it I was so focused on getting the functionality of it that I didn't organize it the best. For example I made 3 classes and a struct and they're all just sitting above main. I could break them out into their own .hpp files to keep it clean. I did use a class based structure for the store and the bank. I think that part is effective and good, however it's not good that they are all in the same place.

I do have some helper functions that I only use in the main of the program. I think that they are alright just above main. I could add more comments to them to ensure that the user of the code knows why they are necessary. Luckily none of these functions are longer than make sense so I don't think I need to break them up.

The supermarket and the bank classes are similar but not enough to do any inheritance, which I'm not even sure how to do in c++ to be honest. The routines in both classes are different enough that making two classes makes sense. The one thing I think that I should change are the variables in the classes to be private. Many, if not all of them, are public which code complete states is a bad idea in most cases. Some of my main methods need a public variable, like the print function for example. But I will comb through to make sure that all that aren't necessary are made private, or I'll make getter functions.

I think reevaluating the global variables I have would be a good idea. To see if I could get rid of them or not because they're not a good idea most of the time. Luckily this program is only ~200 lines so I don't think that it's too big of an issue. But it can't hurt to look over it again.

I definitely don't name variables and routines very well so I'll probably go through the whole project and try to make them more descriptive as well. For example when I make a bank and store object I call them bankSim and storeSim. I think that that's confusing because I have a routine for both called simulate() so it seems redundant to call the object sim.

I think I also have some unused variables in my code. For example in my struct I thought that I would have to keep track of wait times but I don't think I ended up using it in a helpful way. Also looking through that I need to also ensure that all of my variables are initialized. My program is working so that's an indicator that they're initialized at some point but I also have a poor track record with not initializing everything I need to.

One thing that I have always had trouble with is making sure that the code works throughout the refactoring process. I think frequent pushes to GitHub during the refactor would be a good idea as well as testing the program with the same inputs after each change to make sure its functionality is still in tact. I've had to roll back my changes a lot of times to get back to code that worked after refactoring a lot in the past. I should make it even more robust where I make some shell code or change the makefile to test many different inputs to the ensure that they are the same. Maybe outputting to a file and ensuring that the new file is the same as the old one.

So all in all my program could use a major facelift. I think this should give me plenty to change and correct in my original program.