Business Simulator Refactoring Plan

Nicole Morris

I created my program hoping that it could be easily modified to incorporate different business scenarios. The goal of my refactoring is to make it easier understand the code so that it is easier to update the code without damaging the original functioning. With this in mind, my refactoring will include modifications to make the purpose of variables, classes, and functions clearer so that it is easier for myself or someone else to understand the code.

My modification plan:

- Remove unused variables, such as “line_” in the event class.
- Rename variables, functions, and classes whose names names are too long. For example, “arrival_events_queue” is a long name. Is there a simpler yet accurate name I could use for it?
- Rename variables, functions, and classes whose names don’t accurately represent their purpose. For example, I realized that the class named “Event” actually represents a customer and stores a series of events. I would rename the “Event” class to be the “Customer” class so it is better represented.
- Create a global variable to represent how long the business is open. This value is currently represented as a magic number throughout the program which makes it difficult to easily modify the program to run the simulation for different periods of time. As a result, I would change this magic number to be a global variable that I could modify in one or two places so that I don’t have to search for where the magic number is every time I want to test a different variation of the simulation.
- Add comments throughout the program, especially the in the hpp files, to briefly summarize the results of each function. For example, I would add a comment to indicate what analysis is accomplished in the function “analyzeResults()” including the resulting output of the function.

In order to make sure the results work, I will test the compare basic results of the program from before and after refactoring. I will save the results of the original program before I start refactoring. As I complete each refactoring step, I will run the program using the same parameters and compare the results to the original results.