Operating Systems A3 Report

Testing via Debug Text

Testing the application occured in two main parts. Firstly, I created a `Logger` class. This class could be used for printing error messages, general information, and most importantly: debug text. Whenever something important occured, I'd print information using a special method aptly named `debug(String)`. I then added a flag in the logger class called `ENABLE_DEBUG`. Disabling the flag prevents debug text from being sent to the console, which is perfect for submitting the assignment.

This setup allowed me to peek into the state of the application as it ran. Keeping an eye on events as they occured, and watching how certain functions were performing. Being able to disable logging at compile time meant I could return to the assignment at a later date, and still have all my debug text ready to go.

Unit tests with JUnit

The approach described in the previous section was beneficial in allowing me to peek at the state as the application ran. However, I also wanted to know that any refactoring that occurred would not break the entire codebase. So I also added some tests powered by JUnit 4. Since I was using Gradle already to build and manage the assignment, adding tests was as simple as adding JUnit to the list of dependencies.