CMSC 335

Project 3

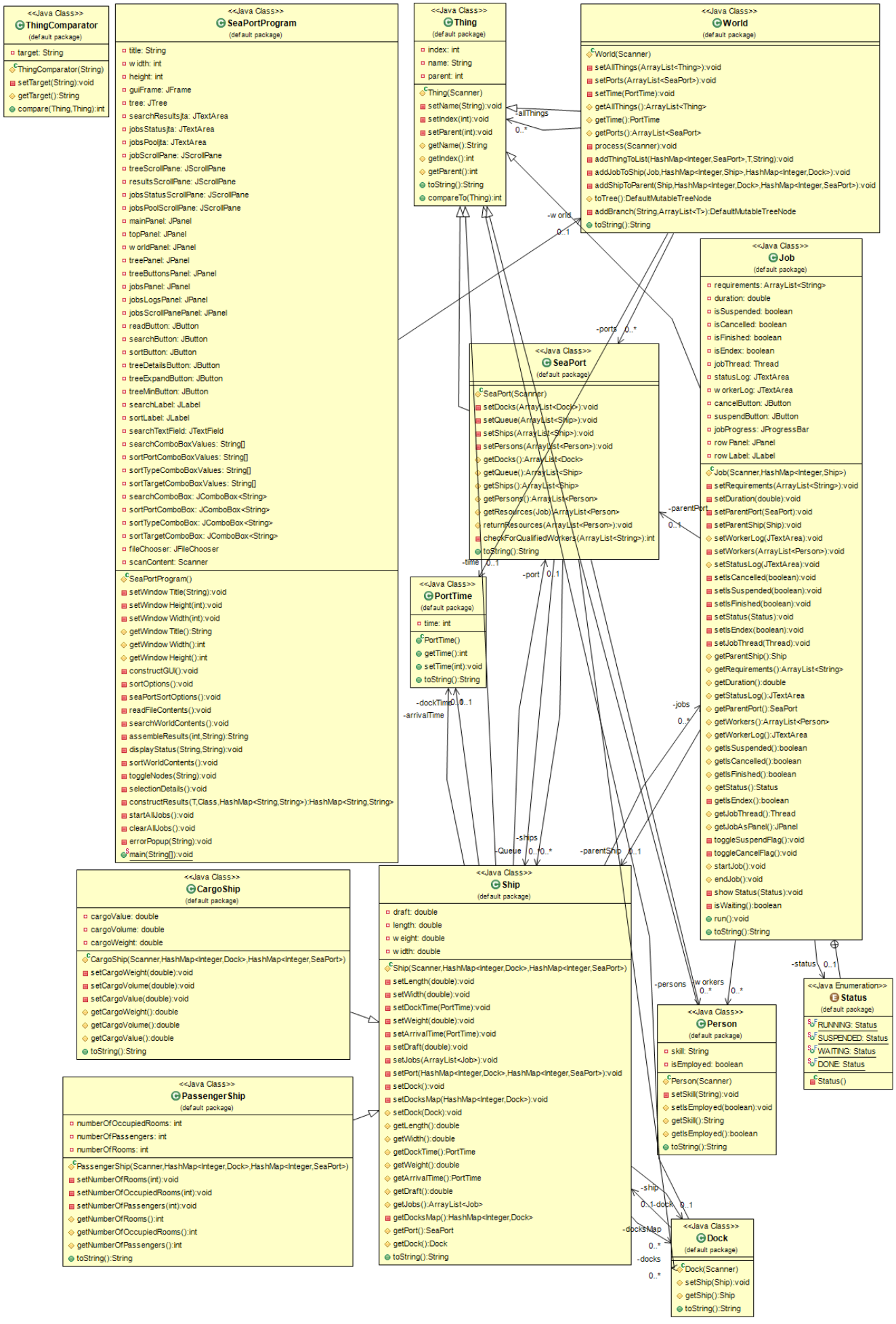
Sea Port Program

Shafro Batyrov

10/14/2018

Shafro Batyrov

UML Diagram:



Shafro Batyrov

Design

This SeaPortProgram consists of many classes and methods that work together as a whole to read input files and display information in a clear and organized GUI. Project 3 is concerned with the specifics of the Job class and the related display of all job objects in the GUI. In line with the focus on concurrent multithreading programming, each job class instance possesses its own related thread which can only gain access to a limited number of dock workers if its associated ship is docked and enough workers of a certain occupation are available. The GUI was subsequently revamped from the previous three-panel design to include a new panel for all currently running jobs, in addition to a pair of status logs denoting the jobs being run and workers currently employed. Progress is displayed via a JProgressBar. Once again, the portTime class was not utilized in this project.

User's Guide

1. Open Command Prompt
2. Change the directory to where these files are saved (i.e cd desktop)
3. Type in “javac SeaPortProgram.java”
4. Type in “java SeaPortProgram”
5. When the GUI pops up, click “Read” button and select the sample data file. (entry is case sensitive and should be spelled correctly. Nontext files will generate an error).

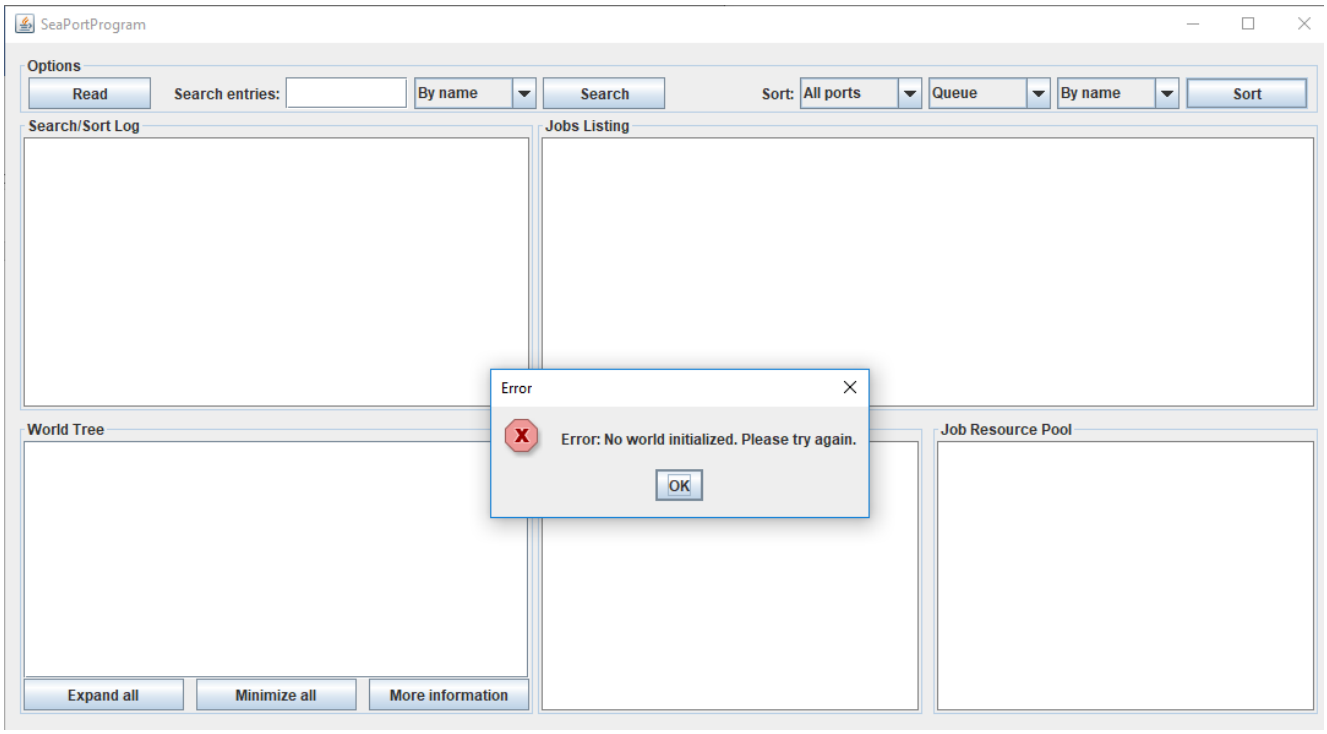
Test Plan

I expect the program’s GUI to look different than it did in Project 2, since I will need more space to display the job status log, resource pool, and job listing panes. I plan to make 3 equally sized job panels and to stack World JTree and the search/sort log in one panel. I also expect to remove the toString() output text when World is clicked. I plan to make 3 buttons: Minimize, Expand, and More Information. They will allow all the information to be expanded and collapsed as needed. If an appropriate node is selected, more information can be found out about it if the “More Information” button is clicked.

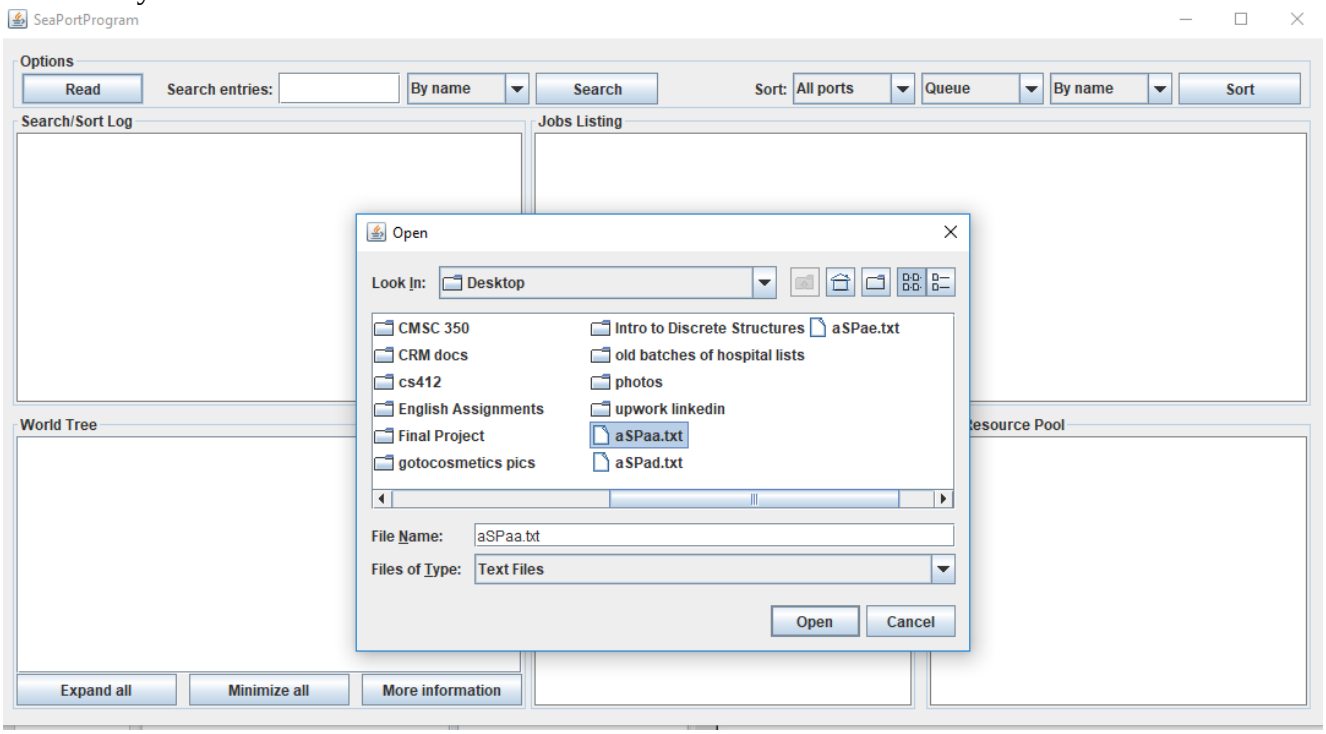
Additionally, I will create a new text file named aSPae.txt. It will be based off of aSPad.txt, but much smaller and easier to keep track of. The original is too big to deal with effectively; there are about a dozen ports and hundreds of ships. With my file, it is easier to chart out the progression of Jobs

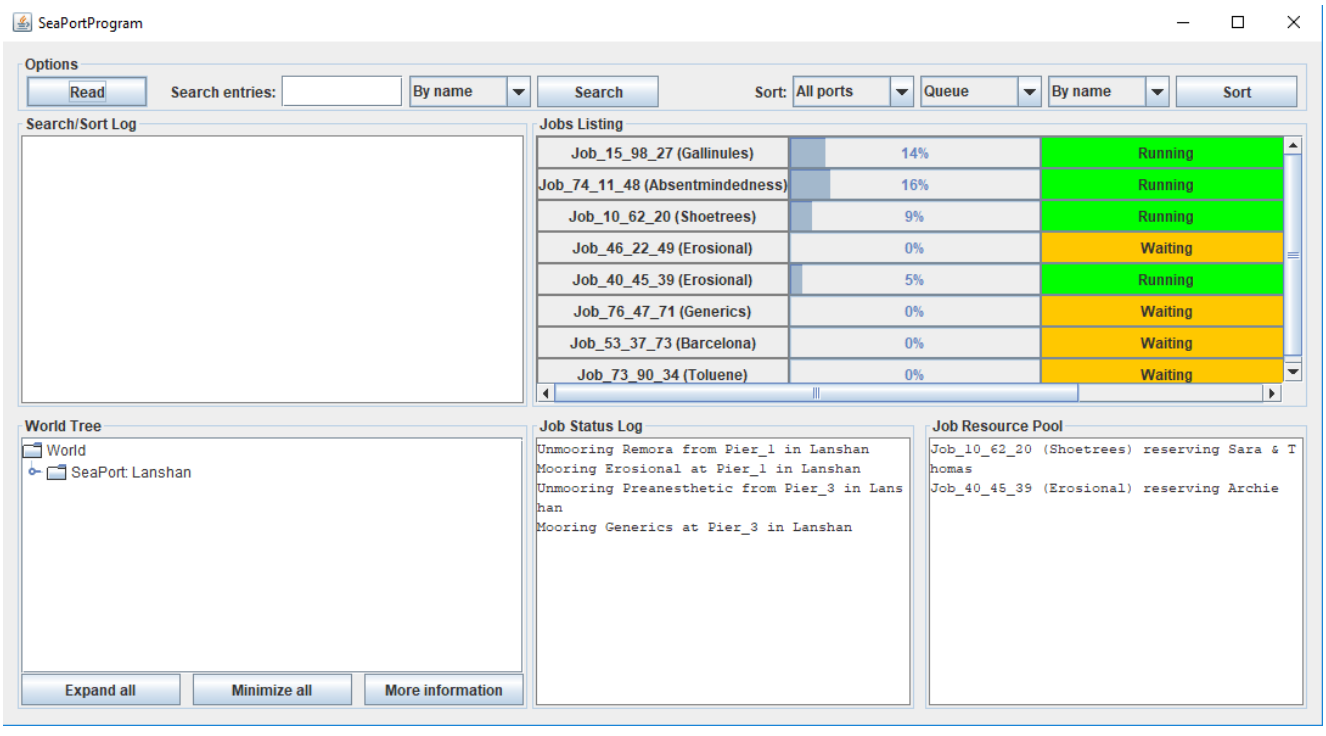
In the following test cases, I will forego testing elements of my program that tested successful in the past 2 projects. These test cases will test whether the new implementations of code are well executed or not.

Selecting the Sort Button with no input: expecting an error message: Success



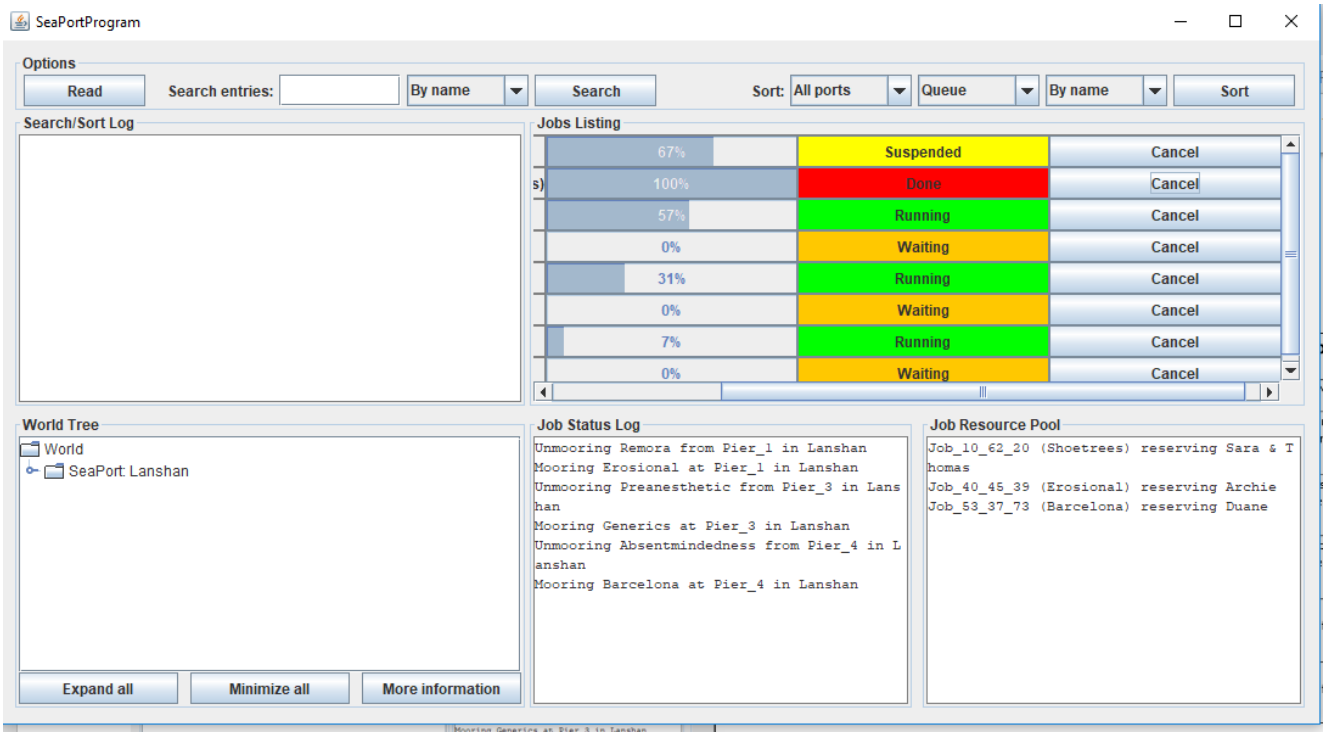
Opening aSPaa.txt: expecting it to compile: Success





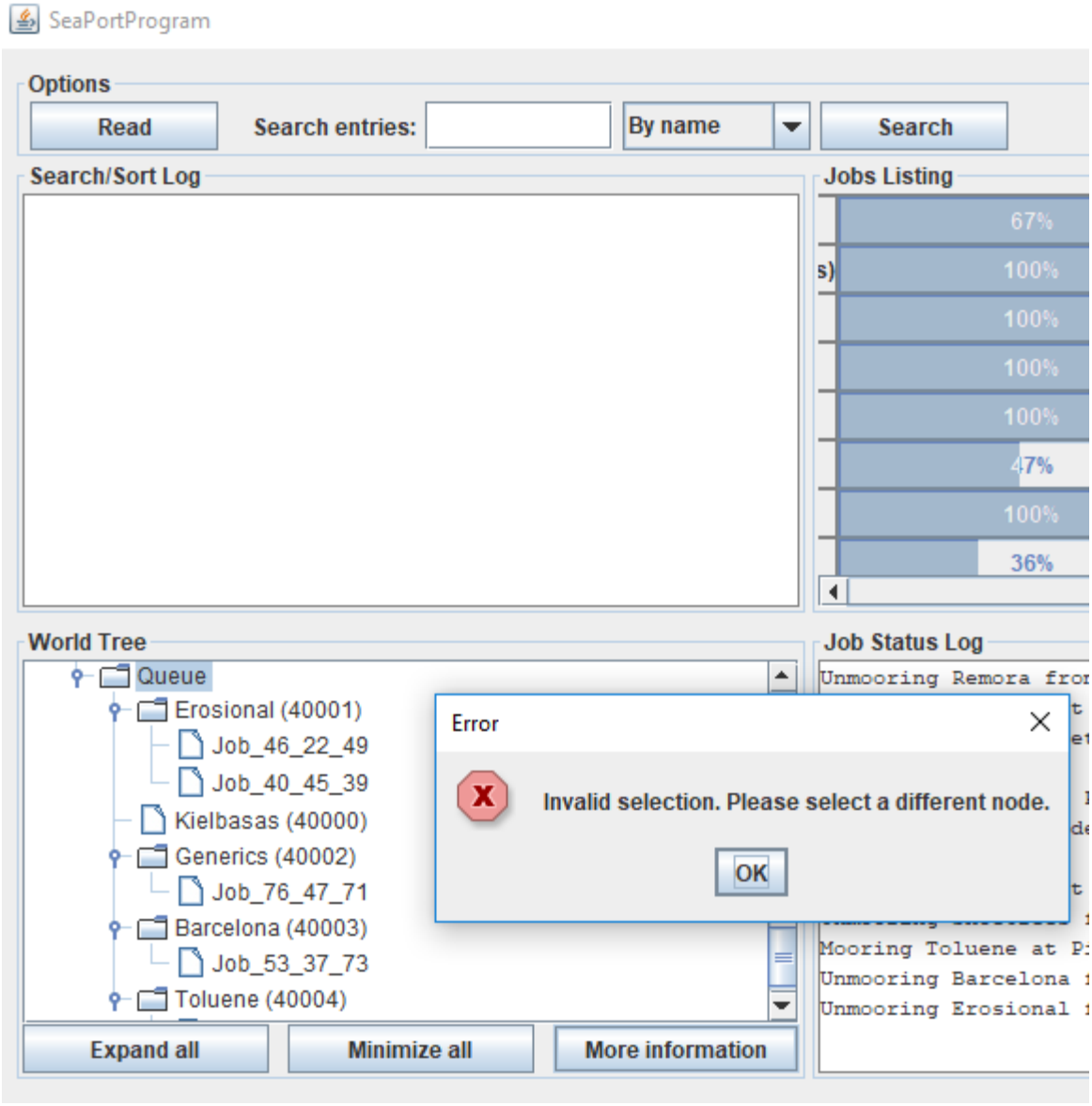
Testing the Suspend and Cancel Buttons: Expecting process to suspend or cancel respectively and change colors: Success

Shafro Batyrov



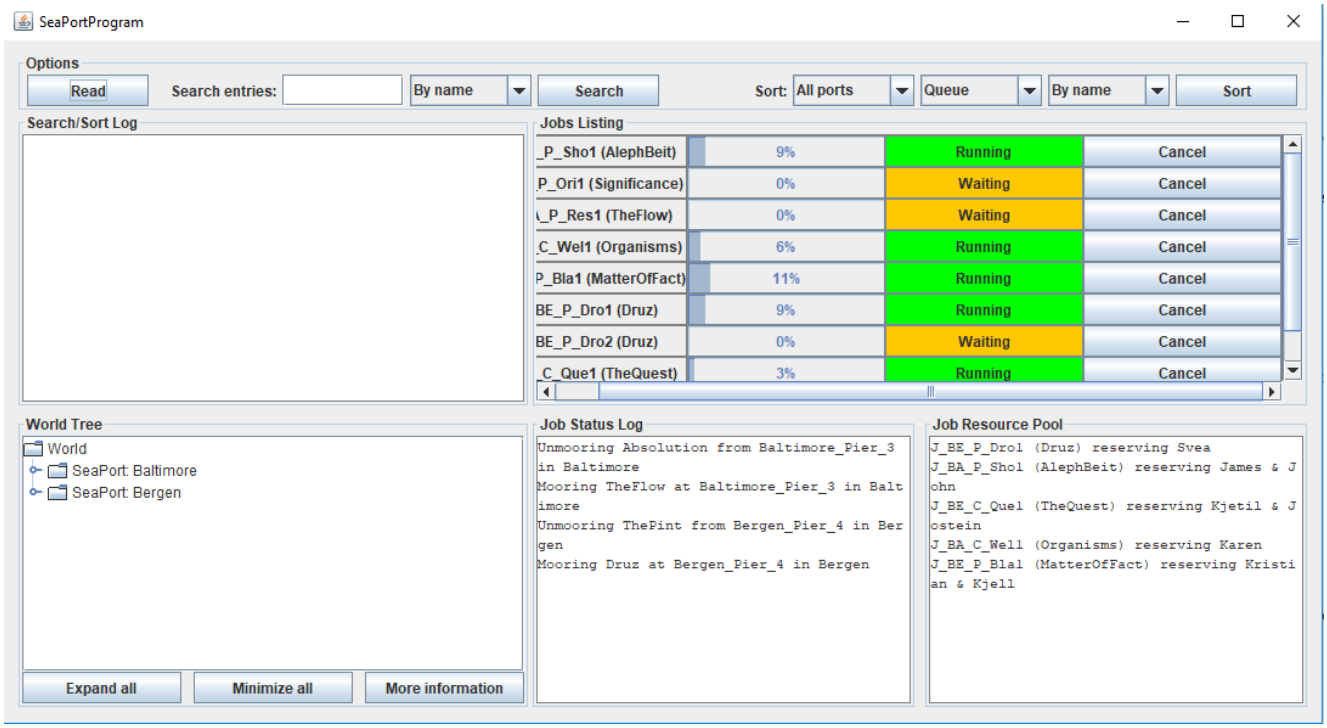
Selecting the Queue node and “More Information”: expecting a error message prompting user to pick a different node: Success

Shafro Batyrov

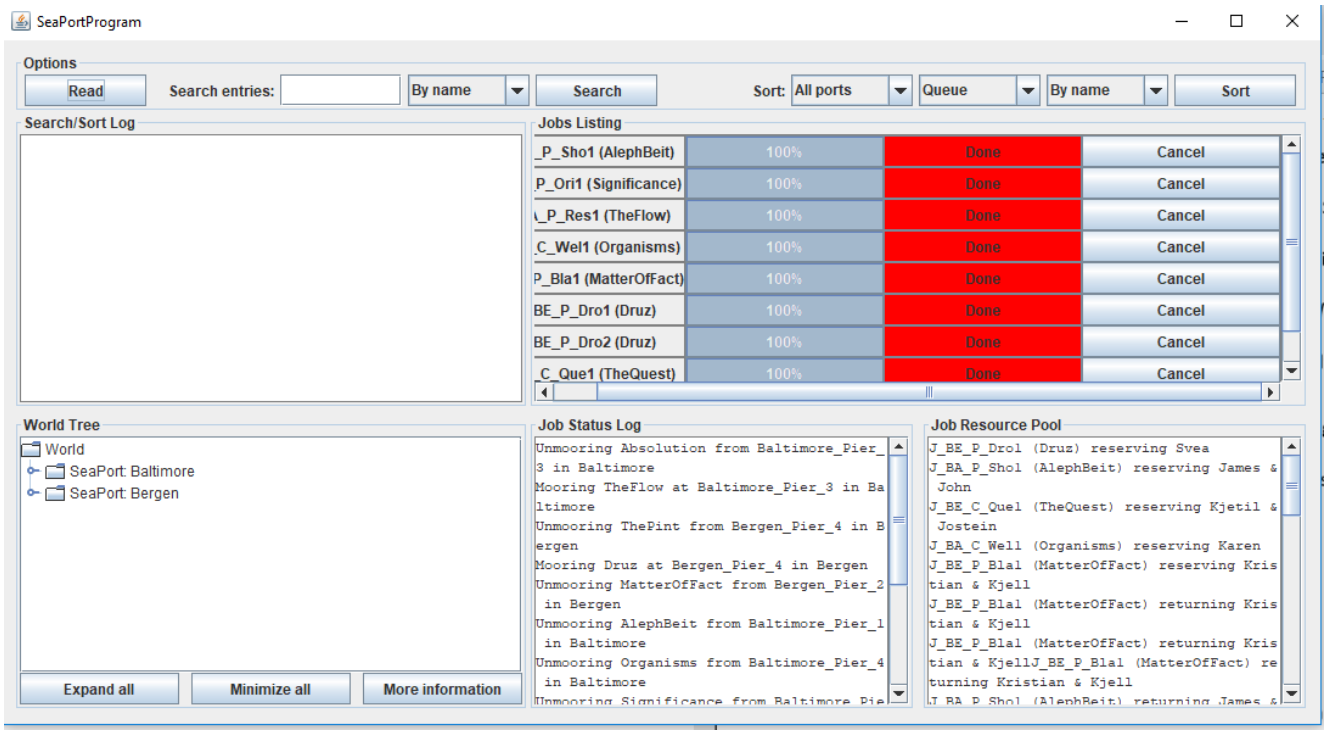


Opening text file aSPae.txt, expecting it to compile and run at different speeds: Success

Shafro Batyrov



All jobs are eventually completed as expected, with the waiting vessels using the workers as they become available in the worker resource pool.



Shafro Batyrov

Lessons Learned -

This project took taught me about synchronization and how to use the JProgessBar effectively. I will say that the GUI was a little complex/overwhelming for me in the beginning. There were so many more components to add than in the previous 2 projects! But, with much research and design errors, I was able to produce a good looking and functional GUI.

On to Project 4!