Name: William Crutchfield

Date: 02/22/19

Course: CMSC 335 – Object-Oriented and Concurrent Programming

Project: 3

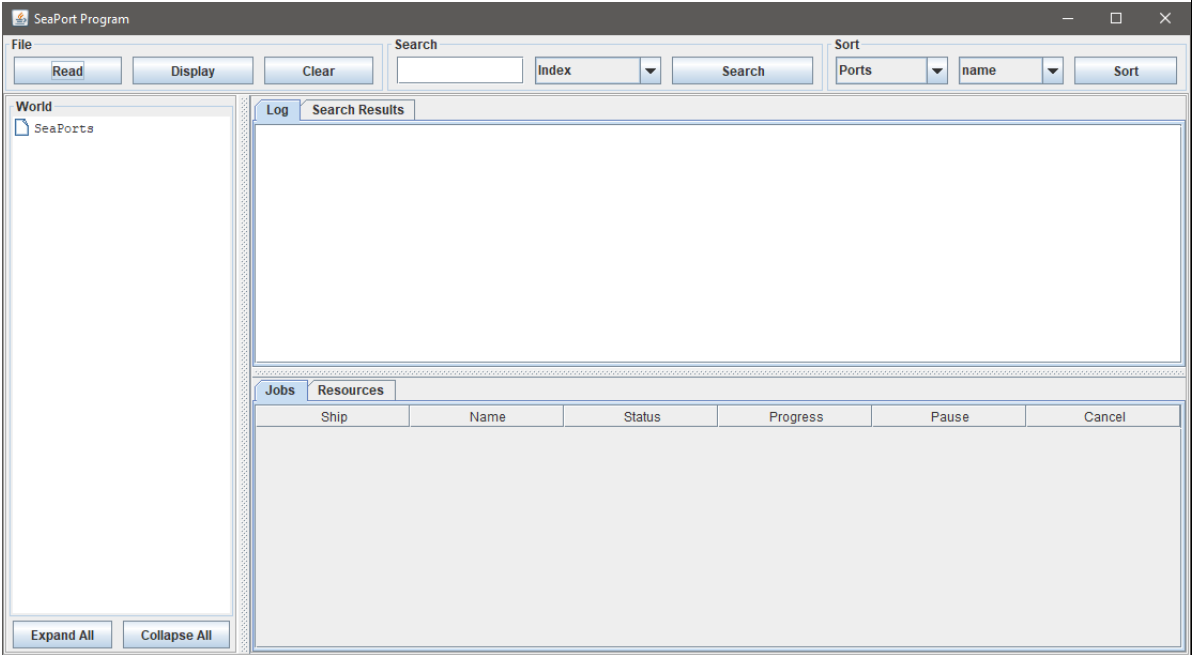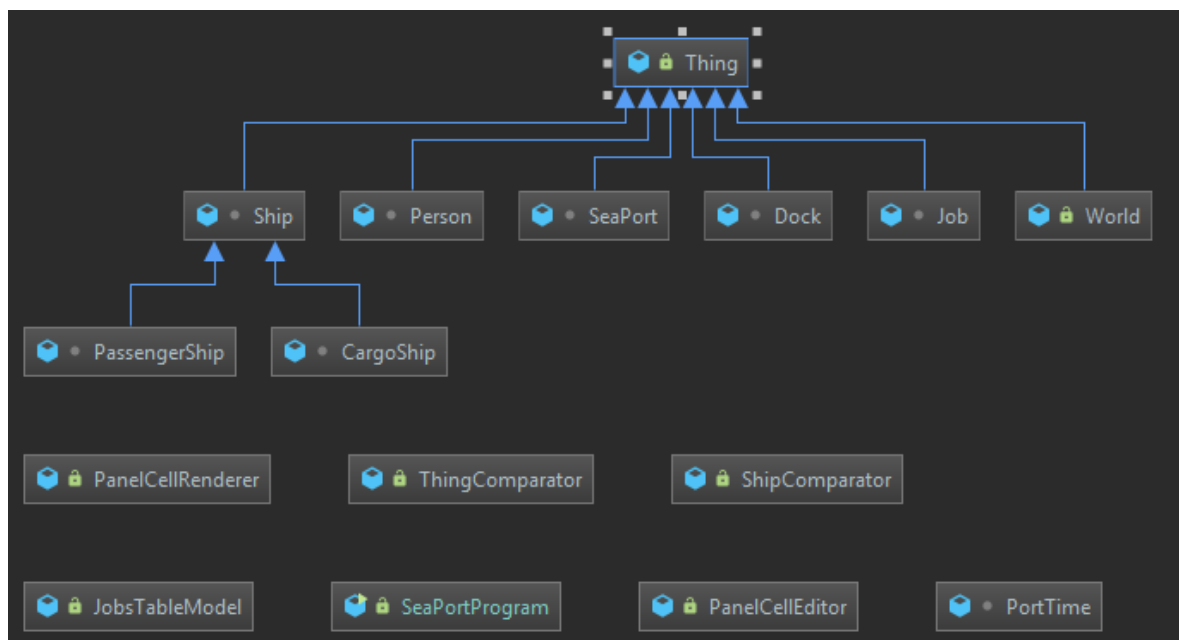| 1	**Source code, data files, and configuration files (if any)** Insert a zipped file of NetBeans ALL project files (so that it could be unzipped and loaded into NetBeans IDE again), zipped file of all data files, and zipped file of configuration files (if any). : SeaPort-Project.zip
2	**Design** Insert here UML Class diagram, explain classes, variables, methods, explain how classes tie to the requirements of the project: The main difference in this UML Diagram compared to the previous project, is that we have 3 new classes, jobsTableModel, PanelCellRenderer, and PanelCellEditor. The jobsTableModel class is custom implementation of TableModel, used to create the jobsTableModel, used in the jobsTable in the SeaPortProgram. The PanelCellRenderer class is a custom implementation of the TableCellRenderer, which is used to render table cells that contain JPanels in the JTable. The PanelCellEditorClass is a custom implementation of the TableCellEditor, which is essentially used to allow buttons within the JTables.

3	**User Guide** Explain how a user starts & runs your project, and any specific features with screenshots: In order to run this project, download the src.zip file and extract. Once extracted, open the src files up in any IDE. Lastly, run the SeaPortProgram.java file.
4	**Test Plan** Complete this table and extend it with your test cases: **Test Case 1:** Test Case: Jobs can be SUSPENDED Selected Input: aSPad.txt – Job_99_51_86 Expected Output: Job_99_51_86 = SUSPENDED Actual Output:
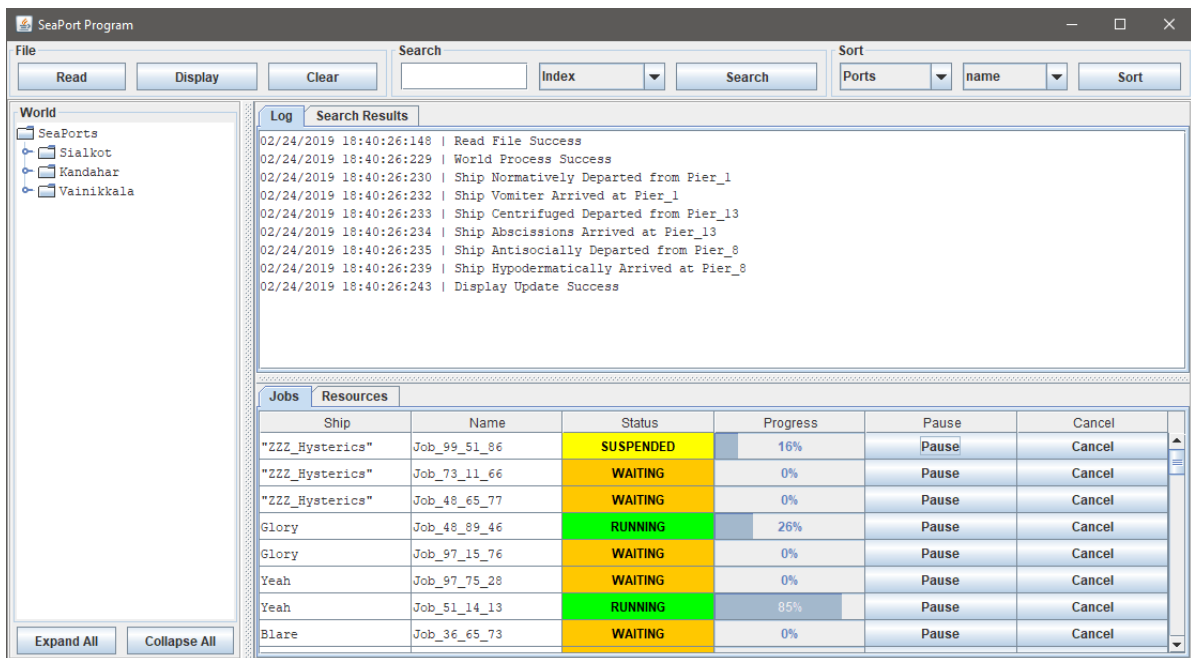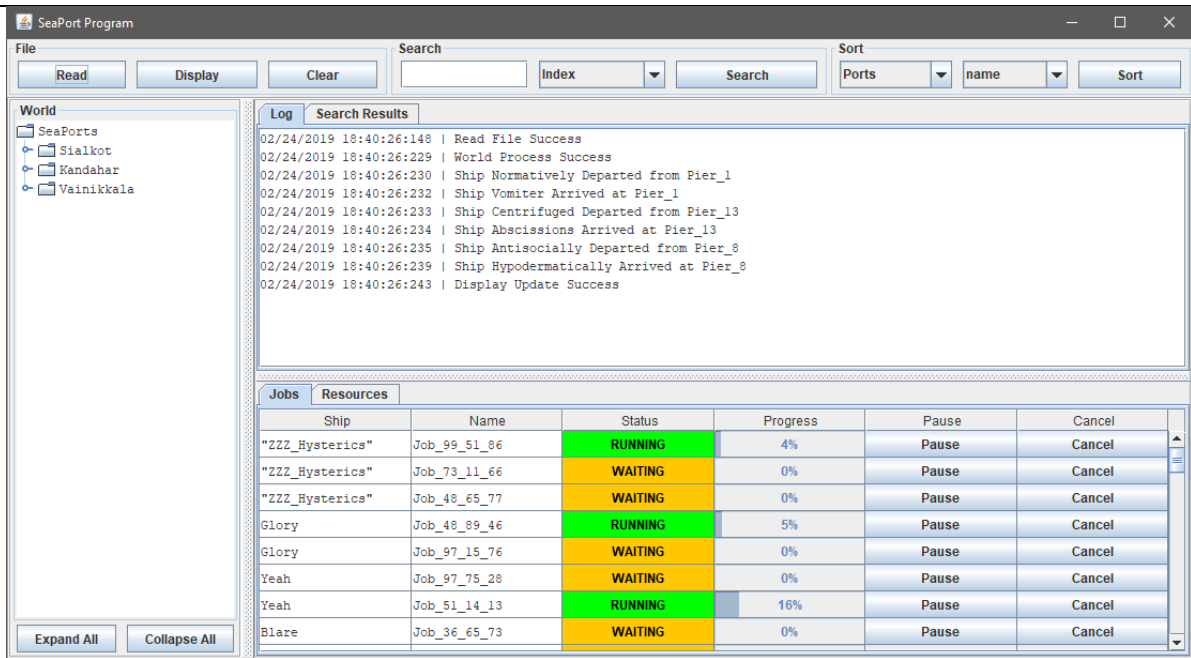
Pass/Fail: Pass, looking at the screenshots, we can see that Job_99_51_86 is
                     SUSPENDED once the Pause button is clicked.
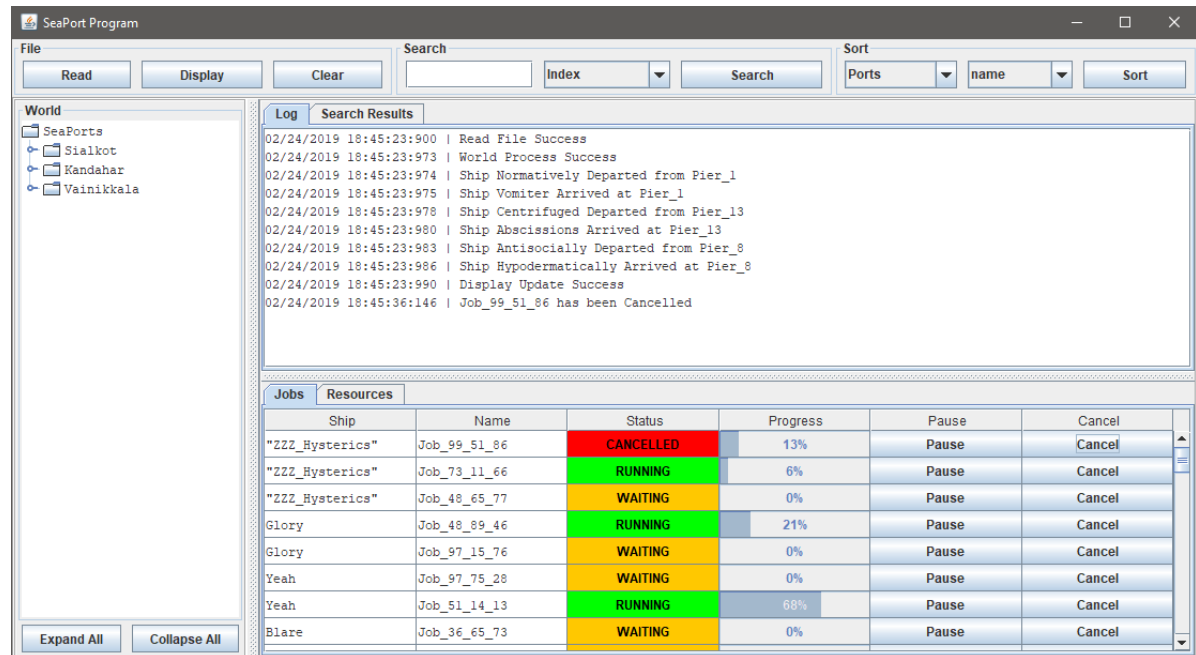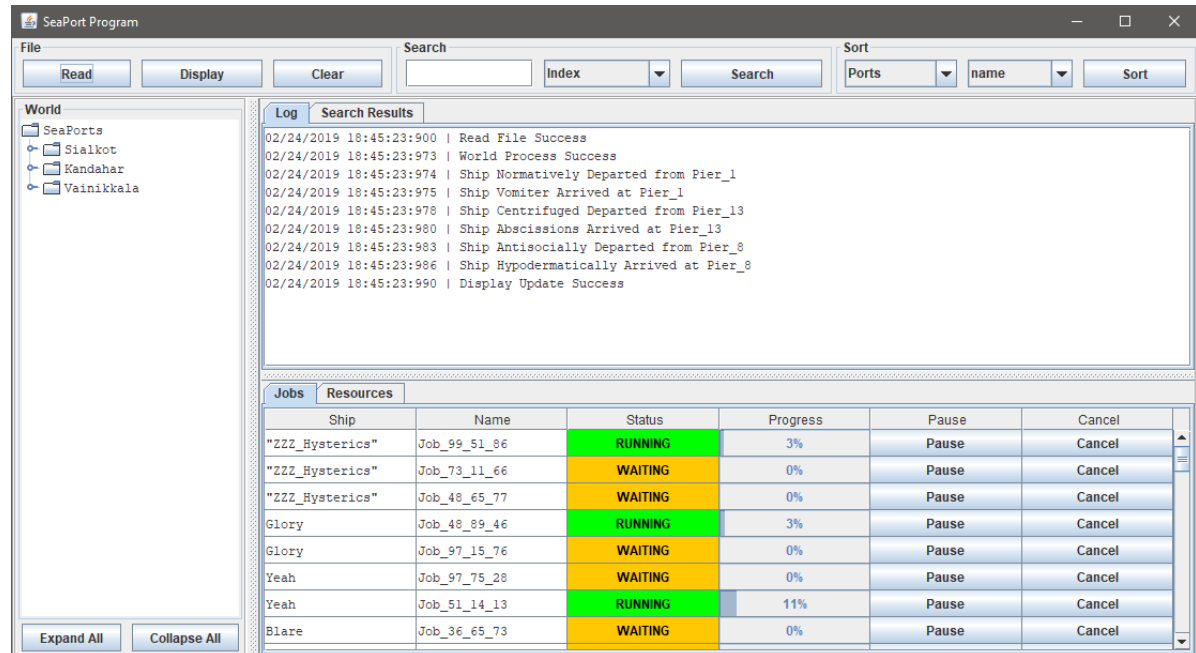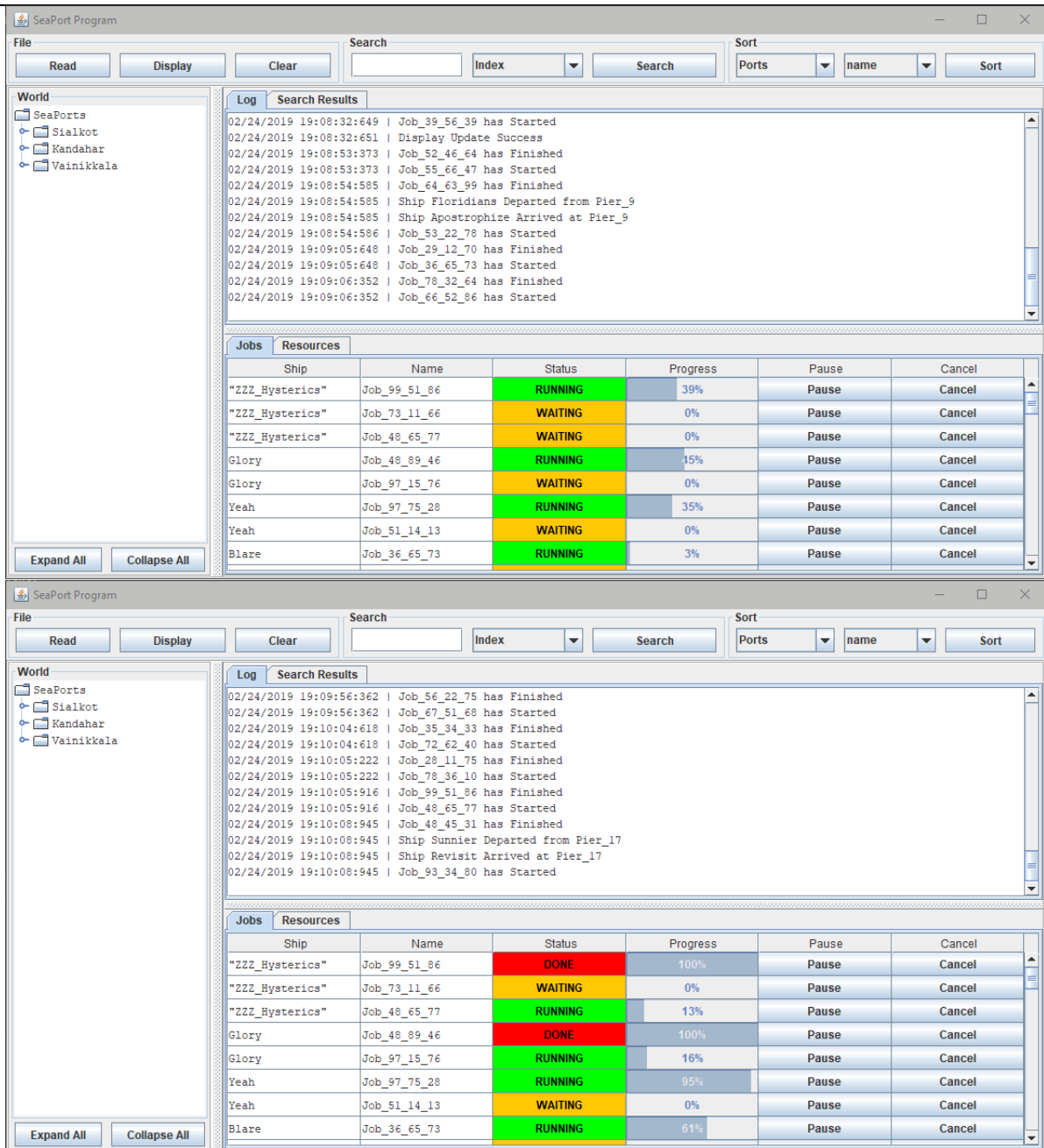
Test Case 2:

Test Case: Jobs can be CANCELLED

Selected Input: aSPad.txt – Job_99_51_86

Expected Output: Job_99_51_86 = CANCELLED

Actual Output:

Pass/Fail: Pass, looking at the screenshots, we can see that Job_99_51_86 is
                          CANCELLED once the Cancel button is clicked.

Test Case 3:

Test Case: Ship starts another Job once the current Job has finished

Selected Input: aSPad.txt – Ship "ZZZ_Hysterics"

Expected Output: Job_99_51_86 = DONE
                                 Job_48_65_77 = RUNNING

Actual Output:

Pass/Fail:	Pass, looking at the screenshots, we can see that once a Job is Finished, another Job within the same Ship starts.

Test Case 4:

Test Case:	Ship Leaves Dock when all Jobs are finished, a Ship from the Queue (that has Jobs) enters the empty Dock.
Selected Input:	aSPad.txt – Ship Floridians & Ship Apostrophize

Expected Output: Ship Floridians Leaves Dock once all Jobs are Finished
                    Ship Apostrophize Enters Dock once it is vacant

Actual Output:

Pass/Fail:	Pass, looking at the screenshots, we can see that the Ship Floridians leaves the Dock once it's Jobs are finished, and Ship Apostrophize enters the Dock once it is vacant.
5	**Reflection and Lessons Learned** Reflect on your experience completing this project and the lessons you learned: Overall, I enjoyed this project a lot. However, I did have quite a difficult time completing this. The biggest issues that I ran into revolving around the JTable. I finally concluded that I needed to create a custom CellRenderer, and CellEditor, in order to create a table with proper GUI elements. I also created the custom TableModel as a convenience for creating the JTable. The other issue I ran into was updating the JProgressBar within the JTable. I ended up passing the instance of the program to the Job in order to deal with these GUI issues. This project provided a great challenge for me, and I learned a lot during the process. Please let me know if there is anything I could improve upon!