Anthony Borza

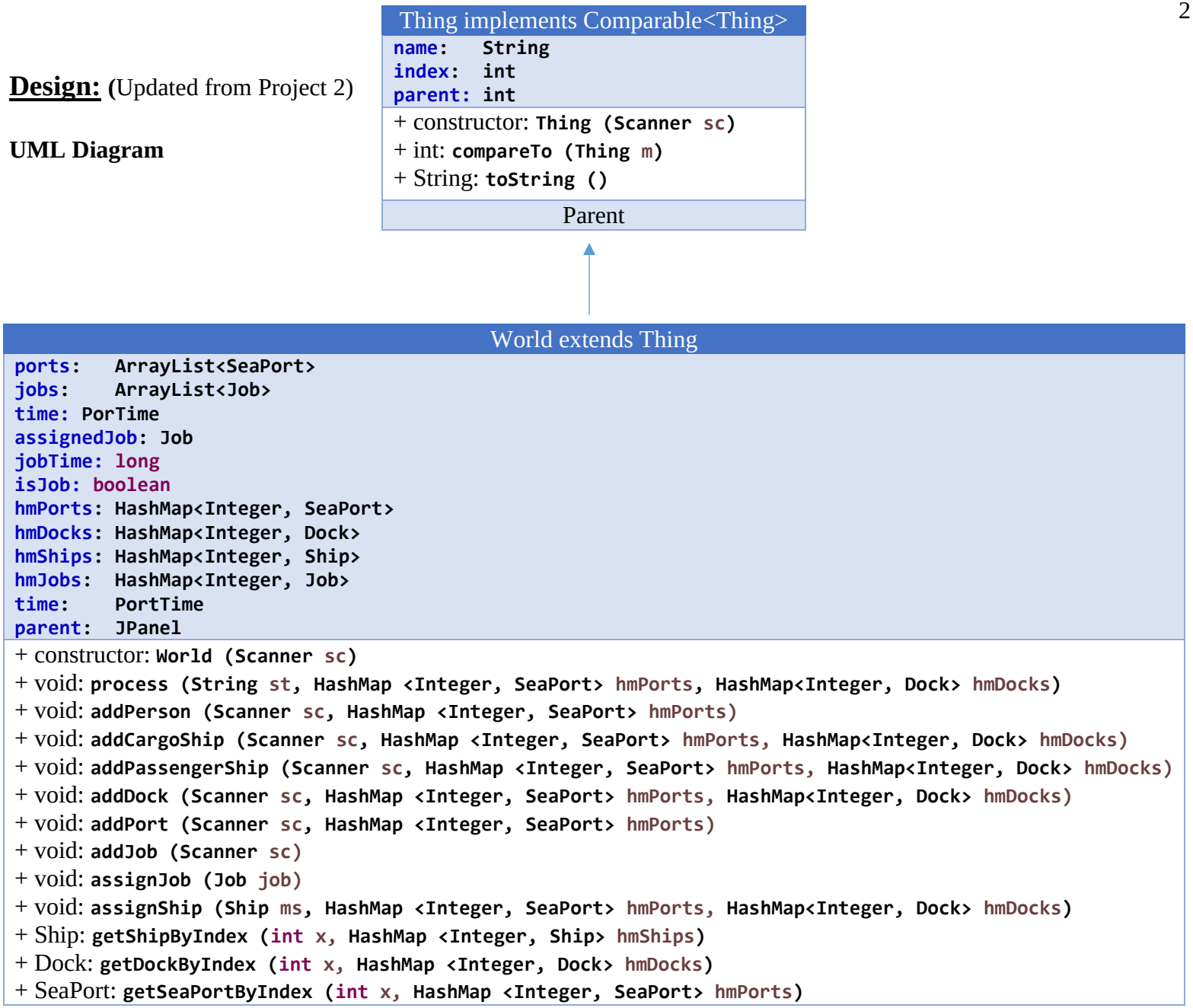
Due Date: 12/04/16

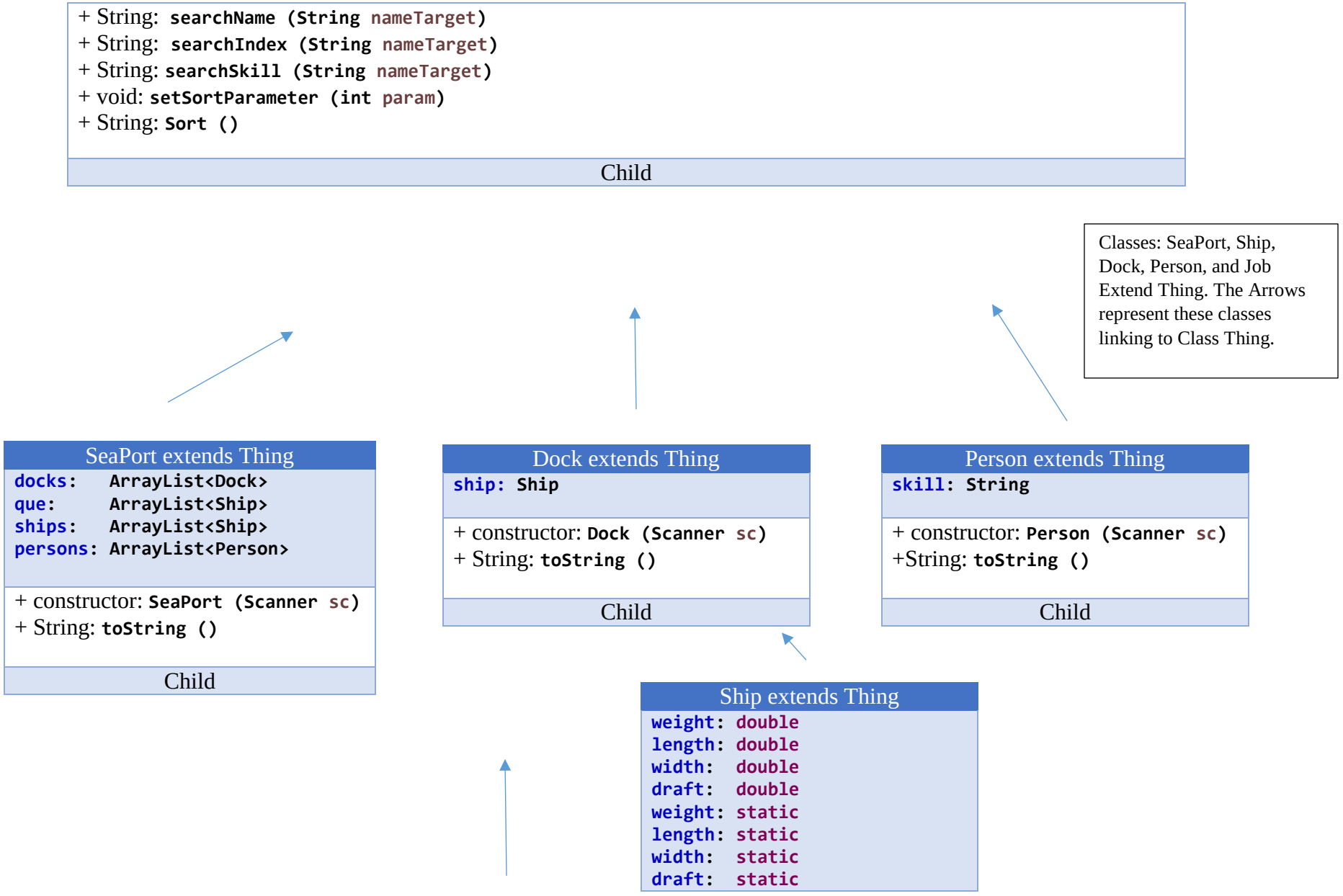
Project 3: SeaPorts: (Extends off Project 2)

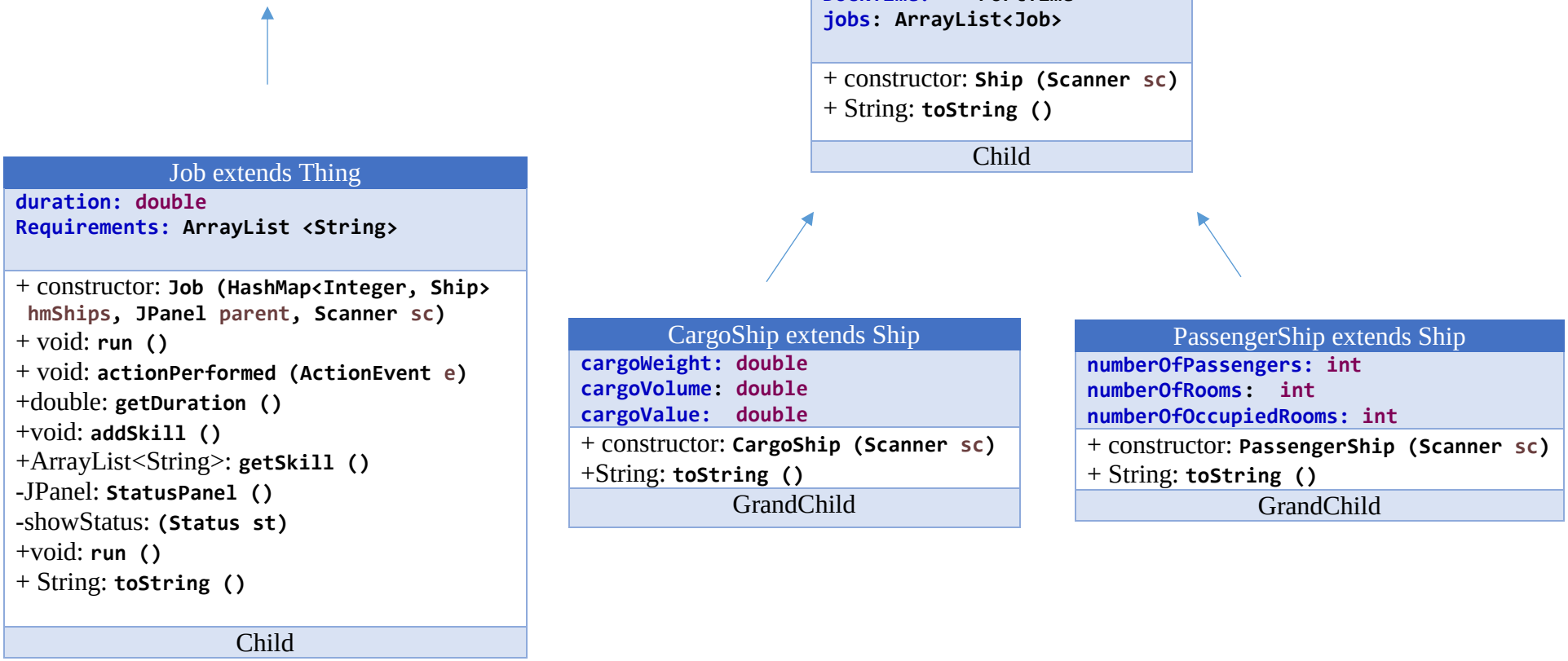
CMSC 335 7980

I. **Design:** (Updated from Project 2)

UML Diagram




+ String: searchName (String nameTarget)
+ String: searchIndex (String nameTarget)
+ String: searchSkill (String nameTarget)
+ void: setSortParameter (int param)
+ String: Sort ()
Child
Classes: SeaPort, Ship, Dock, Person, and Job Extend Thing. The Arrows represent these classes linking to Class Thing.
SeaPort extends Thing
docks: ArrayList<Dock>
que: ArrayList<Ship>
ships: ArrayList<Ship>
persons: ArrayList<Person>
+ constructor: SeaPort (Scanner sc)
+ String: toString ()
Child
Dock extends Thing
ship: Ship
+ constructor: Dock (Scanner sc)
+ String: toString ()
Child
Person extends Thing
skill: String
+ constructor: Person (Scanner sc)
+String: toString ()
Child
Ship extends Thing
weight: double
length: double
width: double
draft: double
weight: static
length: static
width: static
draft: static


ArrivalTime: PortTime
DockTime: PortTime
jobs: ArrayList<Job>
+ constructor: Ship (Scanner sc)
+ String: toString ()
Child
Job extends Thing
duration: double
Requirements: ArrayList <String>
+ constructor: Job (HashMap<Integer, Ship> hmShips, JPanel parent, Scanner sc)
+ void: run ()
+ void: actionPerformed (ActionEvent e)
+double: getDuration ()
+void: addSkill ()
+ArrayList<String>: getSkill ()
-JPanel: StatusPanel ()
-showStatus: (Status st)
+void: run ()
+ String: toString ()
Child
CargoShip extends Ship
cargoWeight: double
cargoVolume: double
cargoValue: double
+ constructor: CargoShip (Scanner sc)
+String: toString ()
GrandChild
PassengerShip extends Ship
numberOfPassengers: int
numberOfRooms: int
numberOfOccupiedRooms: int
+ constructor: PassengerShip (Scanner sc)
+ String: toString ()
GrandChild

SeaPortProgram extends JFrame

```
-JTextArea textArea;
-JTextField searchTargets;
-JLabel searchName;
-JRadioButton name;
-JRadioButton index;
-JRadioButton skill;
-JRadioButton weight;
-JRadioButton length;
-JRadioButton width;
-JRadioButton draft;
-JButton searchButton;
-JButton sortButton;
-JButton cancelButton;
-JButton suspendButton;
-JButton resumeButton;
-JButton readFile;
-JPanel panel1, panel2, panel3, panel4;
-JProgressBar pb
 Scanner sc
 World world
 Job job
```

+ constructor: **SeaPortProgram ()**
+ void: **inputFile ()**
- ActionListener: **Search ()**
- ActionListener: **Sort ()**
+ void: **populateTree (SeaPort port, DefaultMutableTreeNode top)**
+ void: **createNodes (SeaPort port, DefaultMutableTreeNode top)**

PortTime.java

```
time: int
```

+ constructor: **PortTime (int t)**
+ compareTo: **PortTime (PortTime other)**
+ String: **toString ()**

Skill.java

```
name: String
```

+ constructor: **skill (String n)**
+ String: **getName ()**
+ String: **setNmae (String n)**

II. User's Guide: (Updated from Project 2)

- **How would a user start and run the project?**

Steps for using my program:

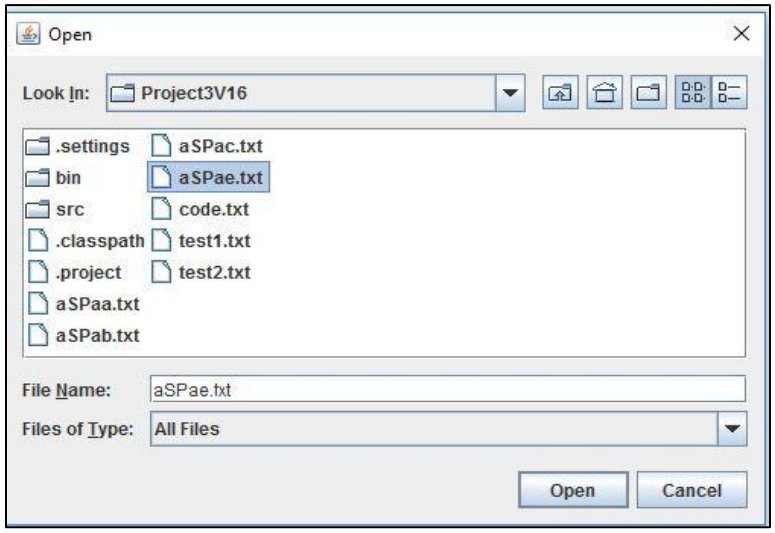
1.) Open a IDE of your choice. (Ex: Eclipse will be used in this Demonstration).

2.) Assuming you have all the files. Open the "SeaPortProgram.java class".

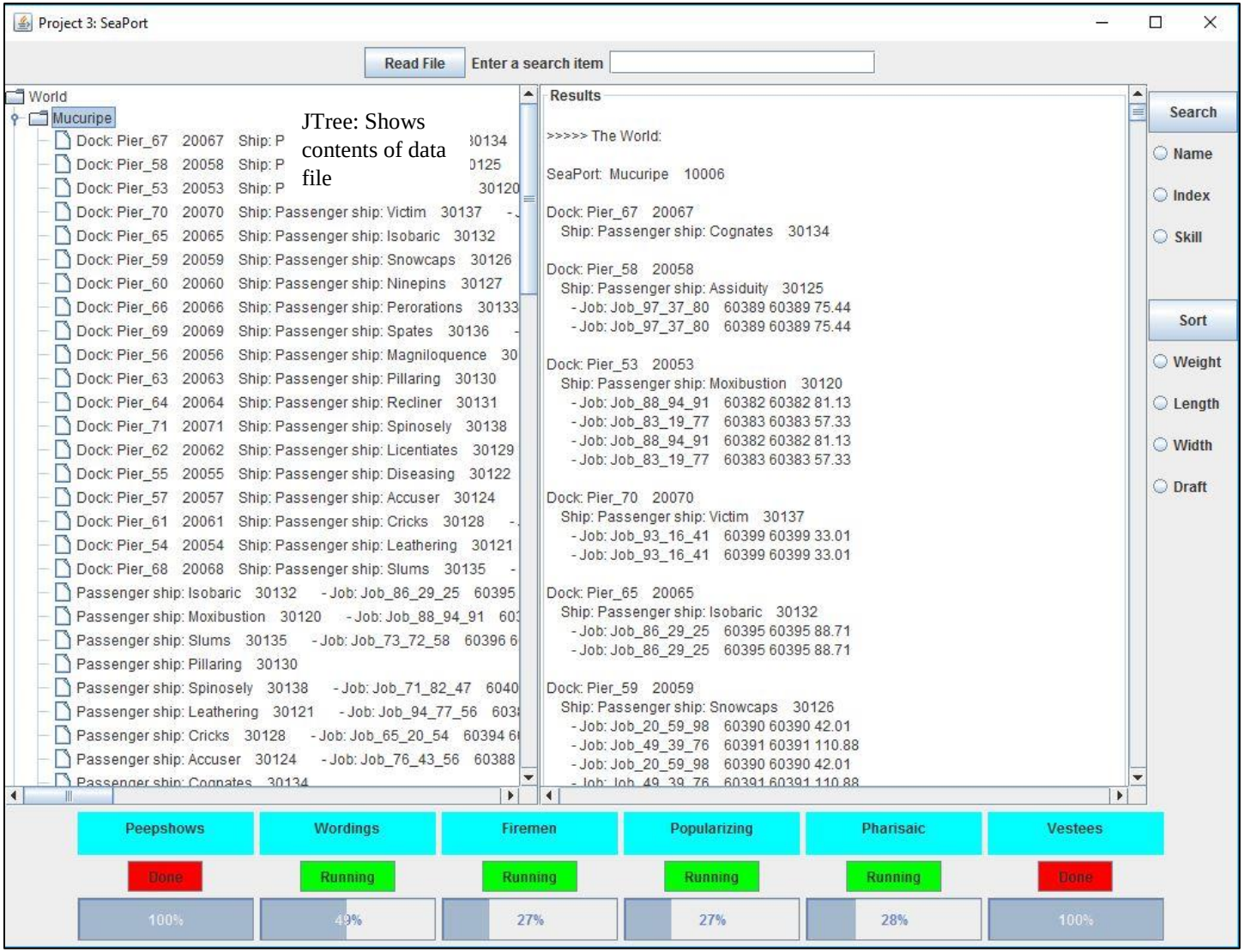
3.) Click on the following button in eclipse to run the program:

 (This can be found at the top-left of eclipse)

4.) You will now be prompted to select a txt file. (Ex. aSPae.txt, will be used)

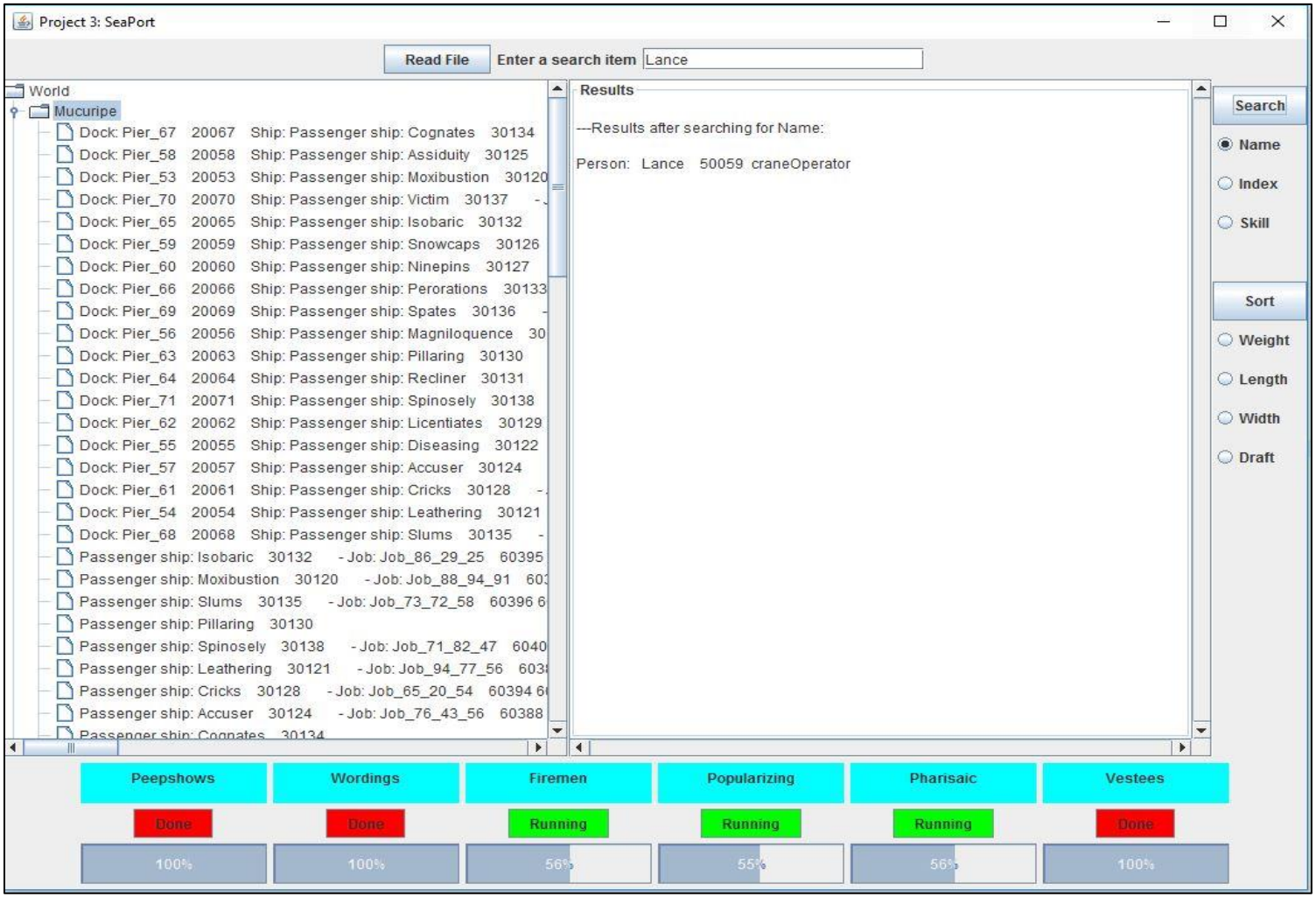


5.) Click the open button, and the following will display.
(**New Features:** JTree, TextArea box for "Jobs in Progress," a running, suspend, done button, and a progress bar)

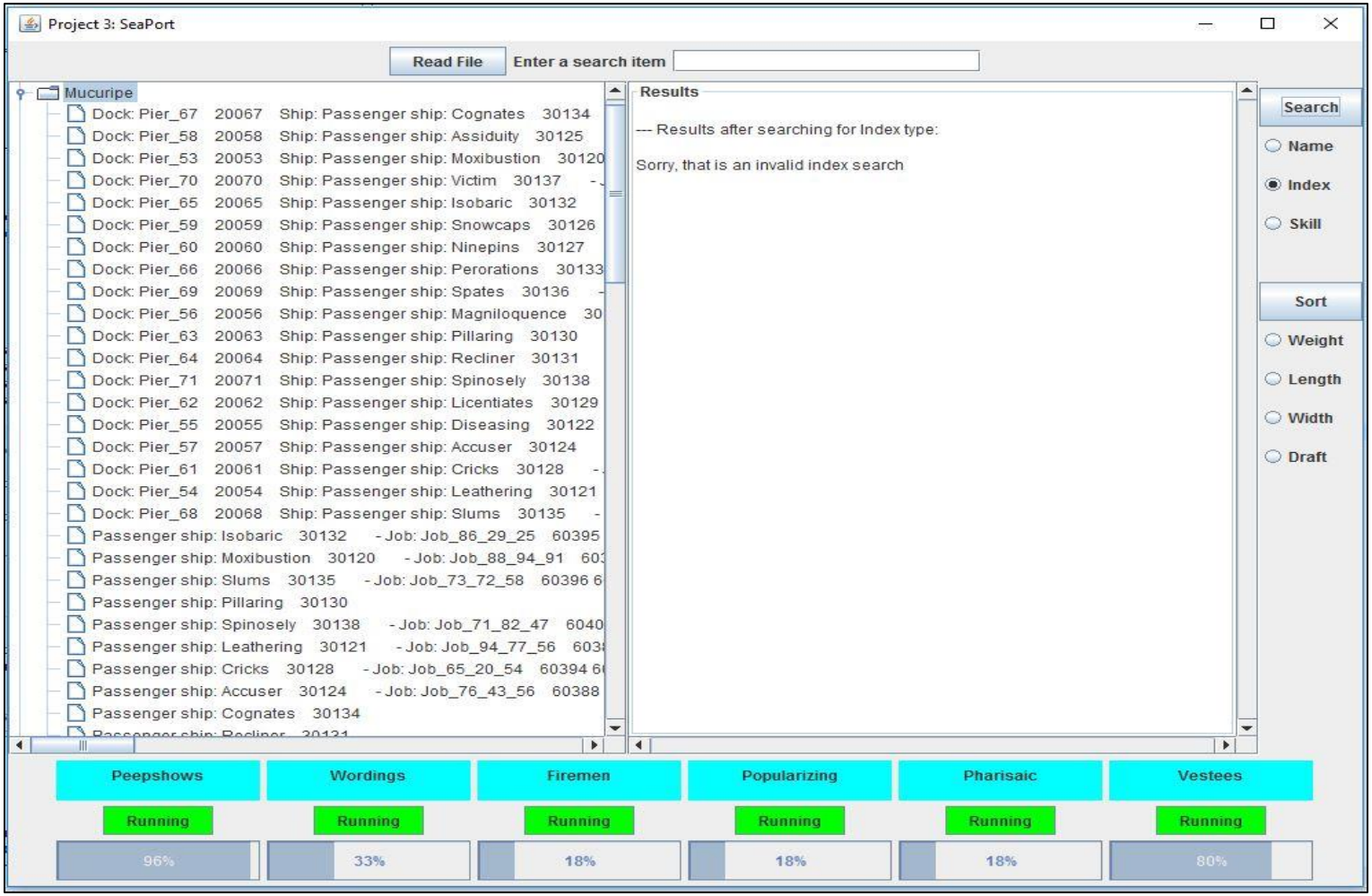


6.) To search for an item in the file, you will have three options: Name, Index, and Skill. (it is not case sensitive)

Name Search Example: Search for Lance

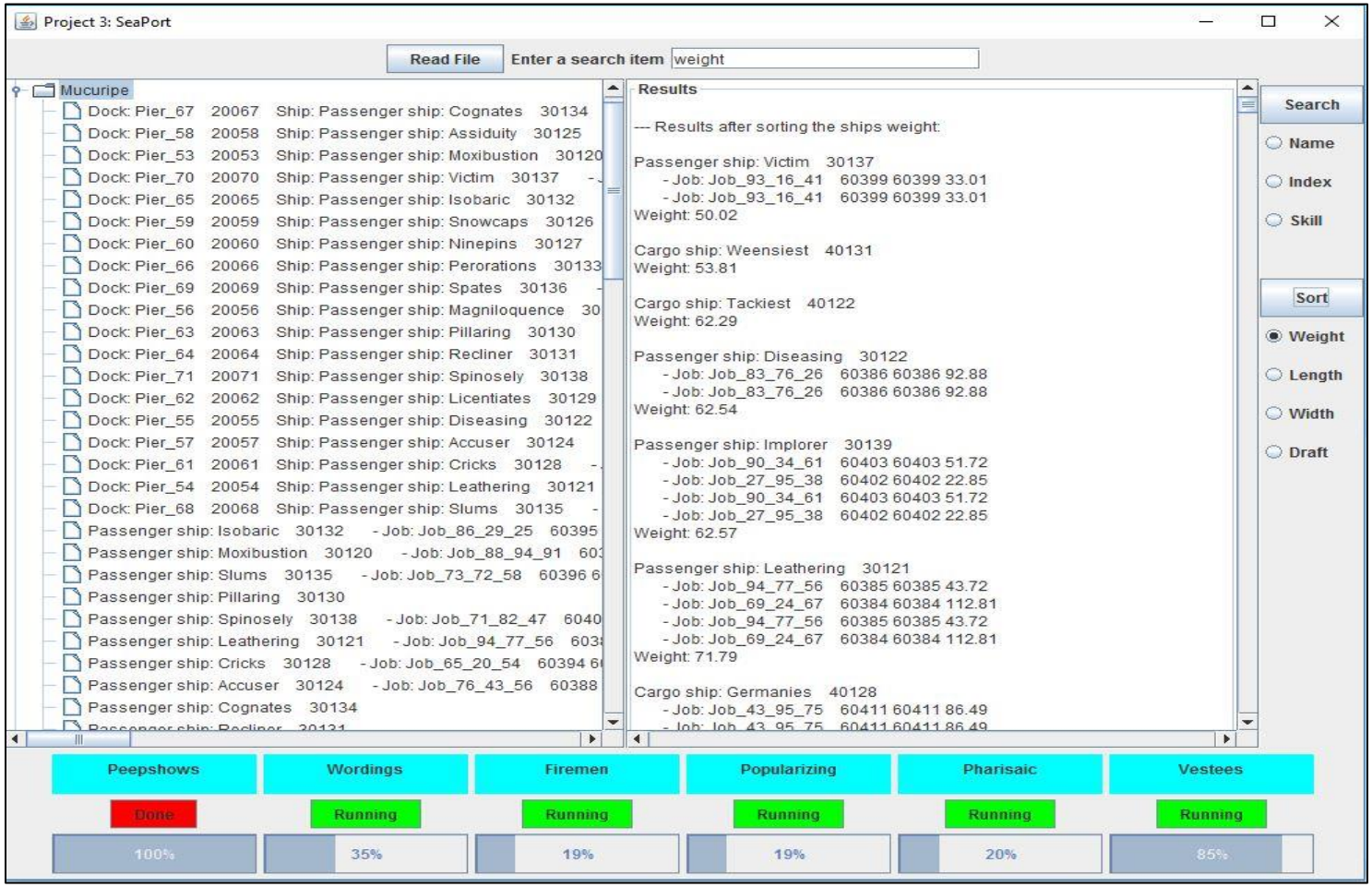


7.) If an invalid search is entered in you will see the following: (Sorry, that is an invalid search. Please try again.)

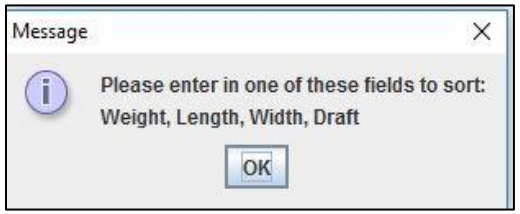


8.) To sort for an item in ascending order you will have three options: Ships Weight, Length, Width, and Draft (it is not case sensitive) **(New step added)**

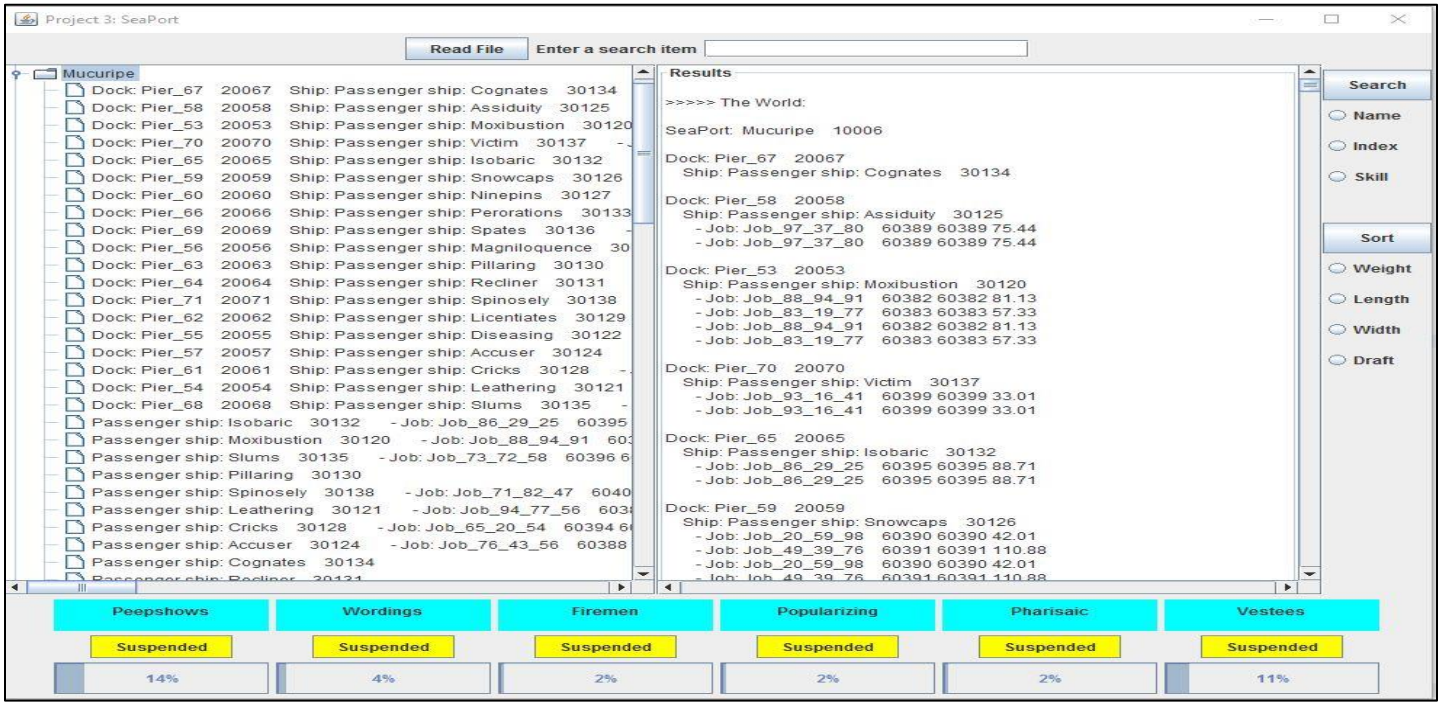
Sort by Ship Weight: Enter in weight



9.) If an invalid sort is entered in you will see the following:



10.) If you want to suspend a job click the suspend button



11.) To open and search another file, select the Read File Button. This will display the same dialog box in step 4.

12.) If you followed these steps correctly, you have successfully used my program the right way.

- **Current Special Features:**
    - User friendly interface
    - Allows you to read in another file without having to run the program again
    - Uses radio buttons to search for the fields in the file
    - Uses radio buttons to sort for the ships weight, length, width, and draft
    - Displays the results nicely
    - Is not case sensitive for what is being search. (Ex: both John, or john, will return the same result).
    - Returns duplicate values for name, index, skill, weight, length, width, and draft (text2.txt contains duplicates)
    - Uses synchronized and multithreading to detect when jobs start and end.
    - Has a suspend, and running button that are colored coded (Ideal for users of this program)
    - Nice JTree panel to show all the nodes for each dock, which has a nice scrollPane.
- **Special Features to come:**
    - A scrollbar around the jobs in progress, so you can see them all
    - Improve GUI

III. **Test Plan:** (Update from Project 2)

- **What do you expect the project to do?**

  Project 3 was an extension off project 2. The first thing I did before extending on to project 2 was to make sure that I understood what the instructions were asking for. The goal of this assignment was to first display the contents of the data file using a JTree effectively using the swing class. I found Oracles documentation as the best resource for understanding how to do this. The next task was to implement threads for each job that represent a task that the ship requires. This will allow us to use synchronize directive, which is necessary for avoid race conditions. By implementing synchronize directive, it ensures that the jobs for that specific ship are being performed one ship at a time, using delays to stimulate the progress of the job. Before starting the project, I knew if I implemented everything properly, then I would get the results I want. The program no displays a JTree of the contents in the data file, as well as, threads for each job that shows the job starting and then finishing.

IV. **Lessons Learned:** (Update from Project 2)

After completing project 3 there are many additional things that were learned from project 1, and 2. Like in project 1, and 2 I realize how important, and difficult it is to work with multiple java classes; as well as implementing threads and synchronization. This project was extremely difficult for me, and I am happy with my results. I need to work on how the GUI displays by adding a JScrollPane, but everything works as it should. I understood and successfully implemented a JTree. I know understand how threads work, as you will see in my program, I was successfully able to show each job starting, and then finishing using a progress bar. This project took the whole week, and a total of 8 hours a day. I hope to improve on this project, so that the final project suites my expectations. Given the circumstances, I do feel as if I am learning a lot, and look forward to presenting you with a well-developed program at the end of the class.