

Anthony Borza

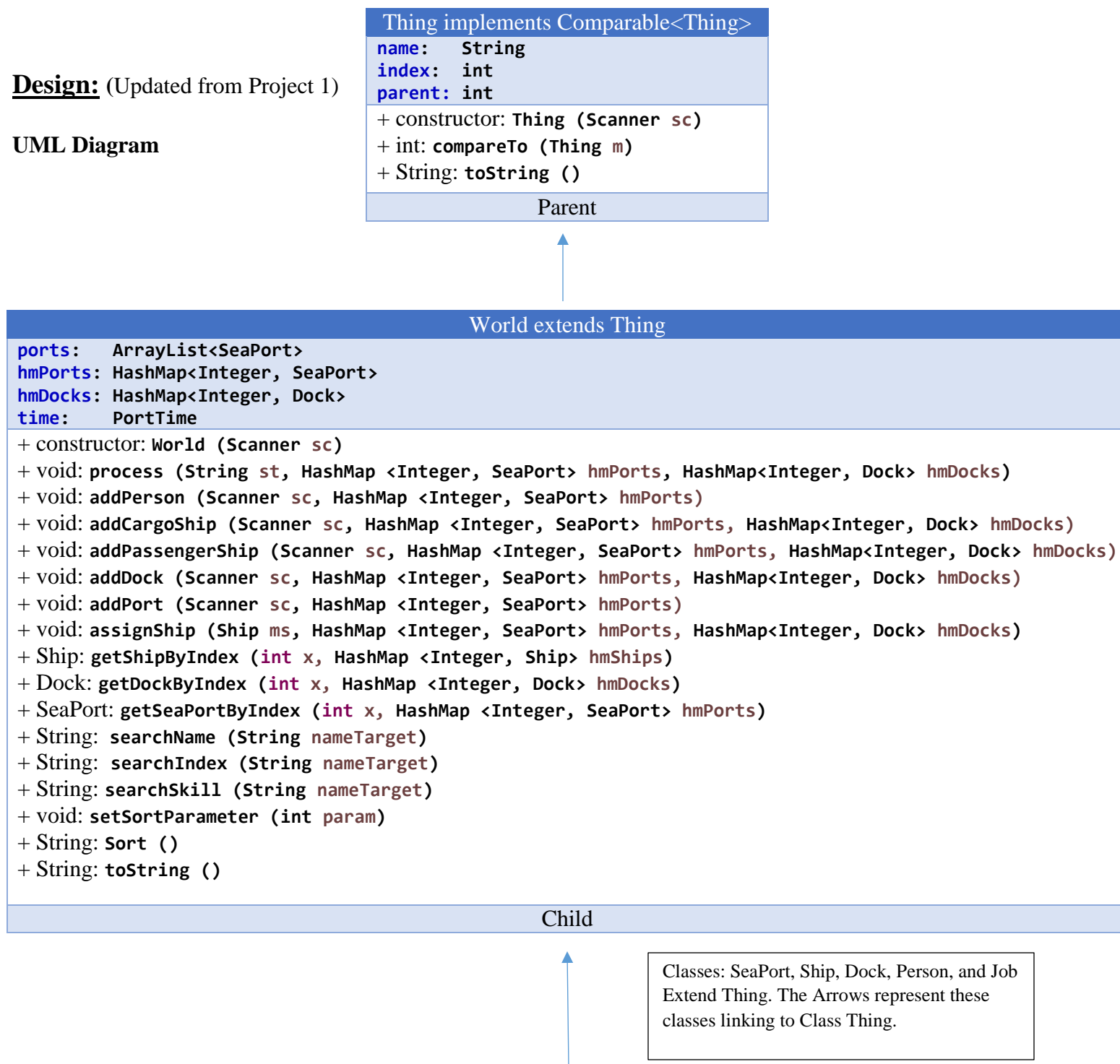
Due Date: 11/20/16

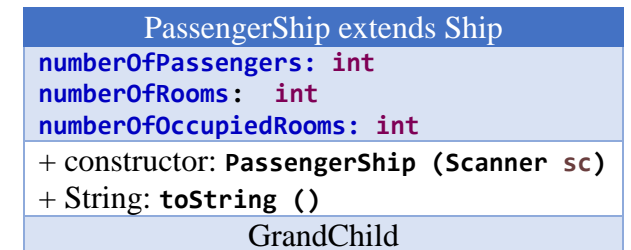
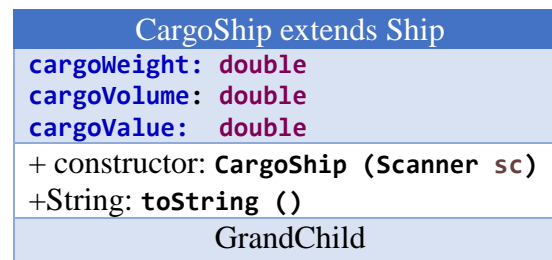
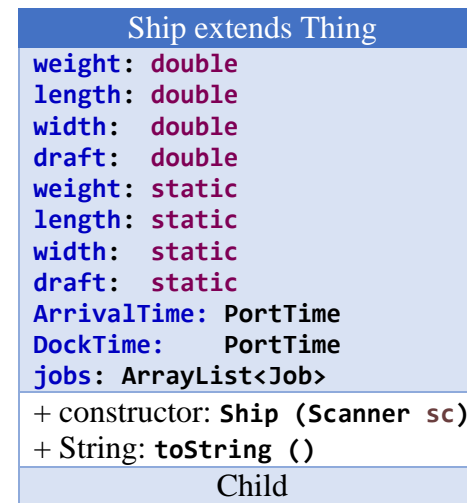
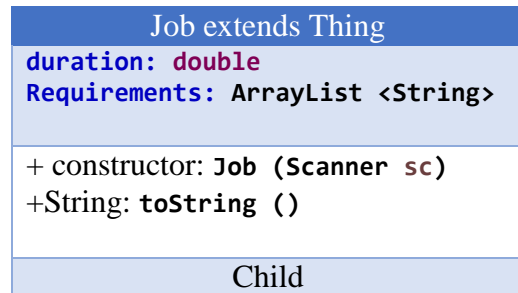
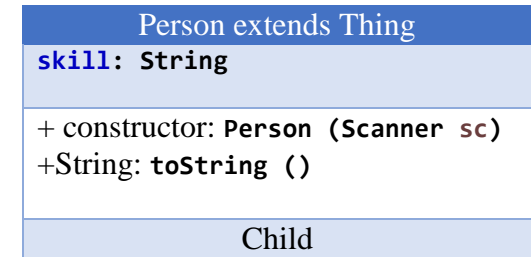
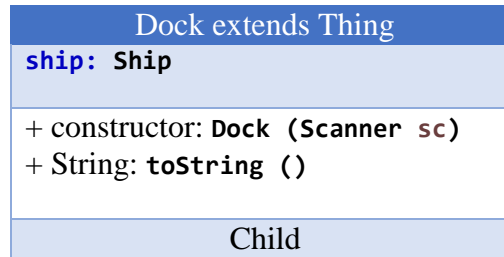
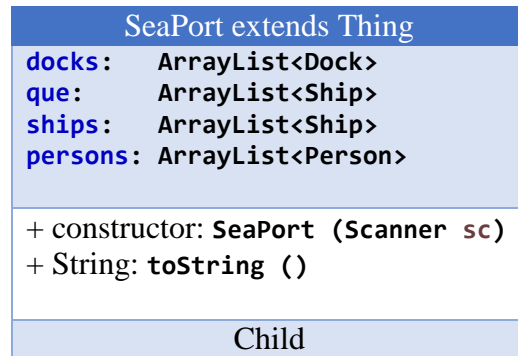
Project 2: SeaPorts: (Extends off Project 1)

CMSC 335 7980

I. Design: (Updated from Project 1)

UML Diagram





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 readFile;
-JPanel panel1, panel2;

+ constructor: SeaPortProgram ()
+ void: inputFile ()
- ActionListener: Search ()
- ActionListener: Search ()

```

PortTime.java

```

time: int

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

```

II. User's Guide: (Updated from Project 1)

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

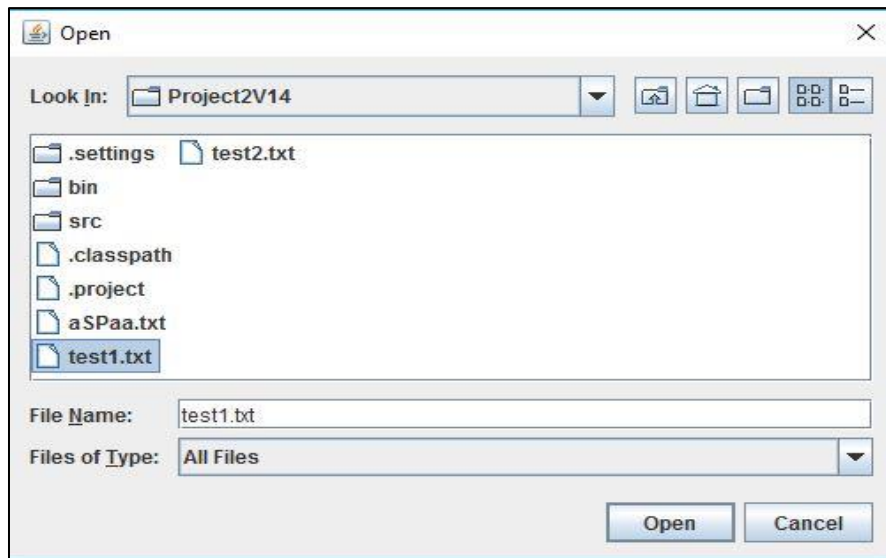
Steps for using my program:

- 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:

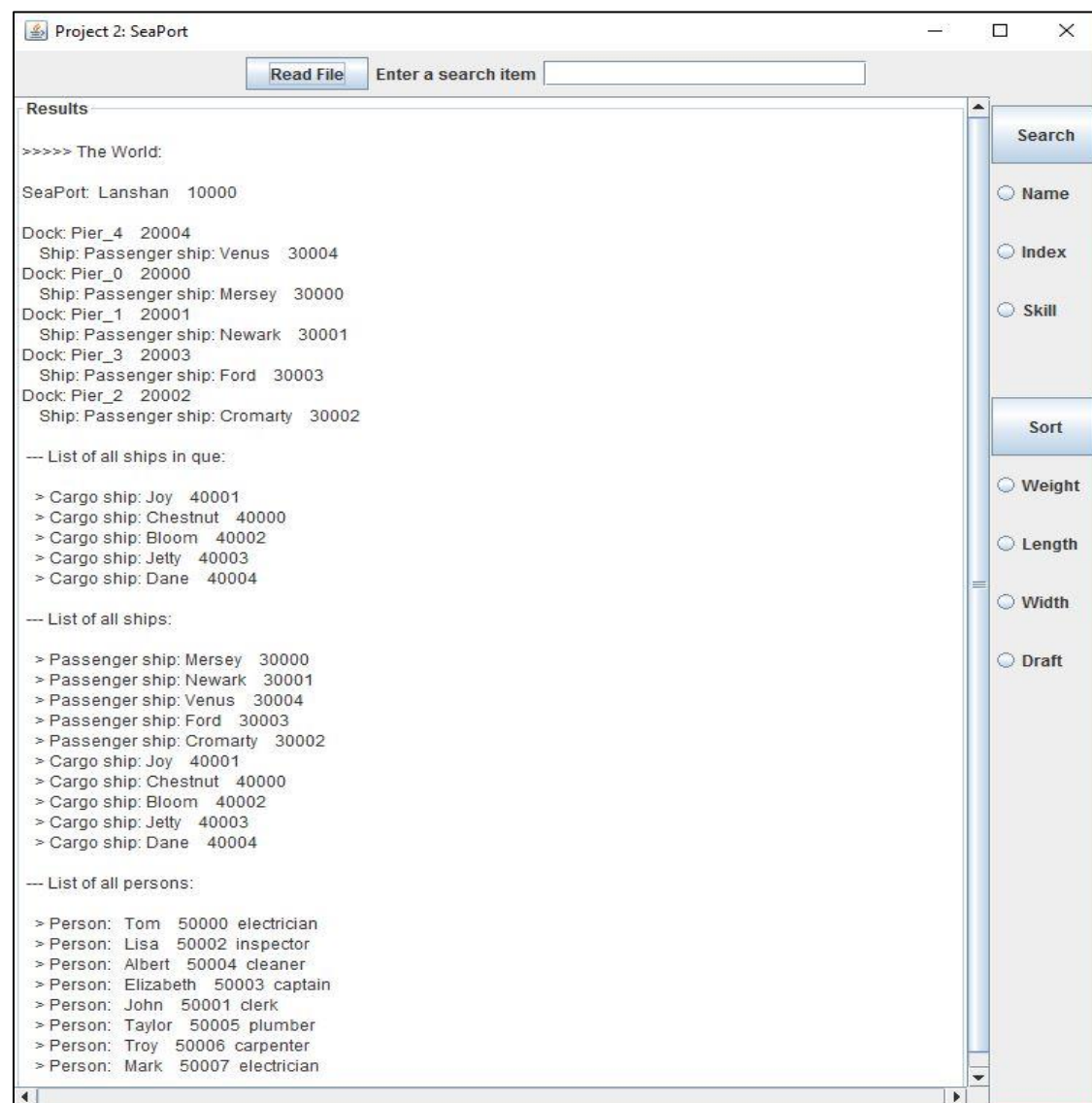


(This can be found at the top-left of eclipse)

- 4.) You will now be prompted to select a txt file. (Ex. test1.txt, will be used)

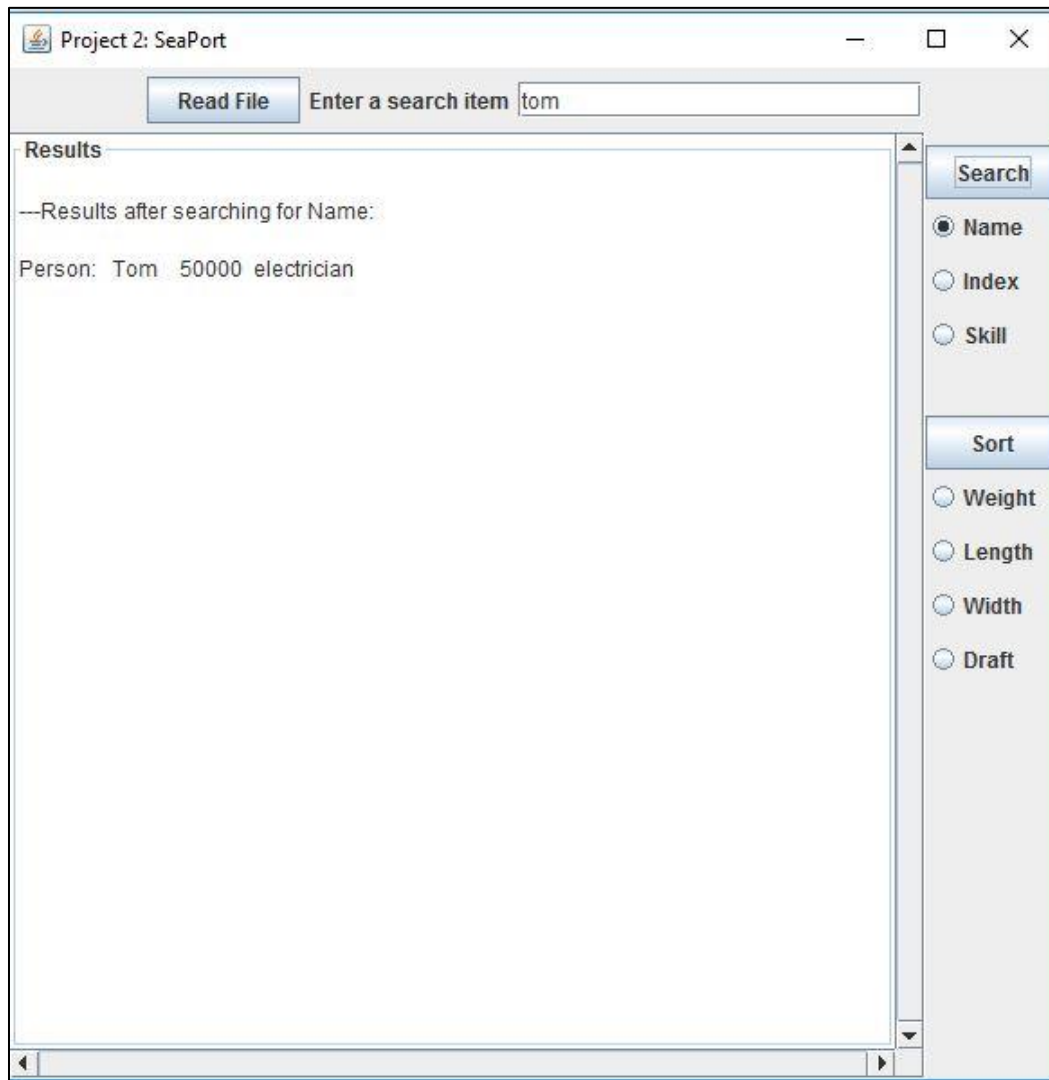


5.) Click the open button, and the following will display:



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 Tom



Index Search Example: Search for 30004

The screenshot shows a window titled "Project 2: SeaPort". At the top, there is a "Read File" button and a text input field labeled "Enter a search item" containing the value "30004". Below this is a "Results" section with a scrollable area. The text inside the results area reads: "— Results after searching for Index type:" followed by "Passenger ship: Venus 30004". On the right side of the window, there is a "Search" button and three radio buttons: "Name", "Index" (which is selected), and "Skill". Below these is a "Sort" button and four radio buttons: "Weight", "Length", "Width", and "Draft". The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

Skill Search Example: Search for electrician

The screenshot shows a window titled "Project 2: SeaPort". At the top, there is a "Read File" button and a text input field labeled "Enter a search item" containing the word "electrician". Below this is a "Results" section with a scrollable area. The text in the results area reads: "— Results after searching for Skill type:" followed by two lines: "Person: Tom 50000 electrician" and "Person: Mark 50007 electrician". To the right of the results area is a vertical sidebar. It contains a "Search" button, three radio buttons labeled "Name", "Index", and "Skill" (with "Skill" selected), a "Sort" button, and four more radio buttons labeled "Weight", "Length", "Width", and "Draft".

Project 2: SeaPort

Read File Enter a search item electrician

Results

— Results after searching for Skill type:

Person: Tom 50000 electrician
Person: Mark 50007 electrician

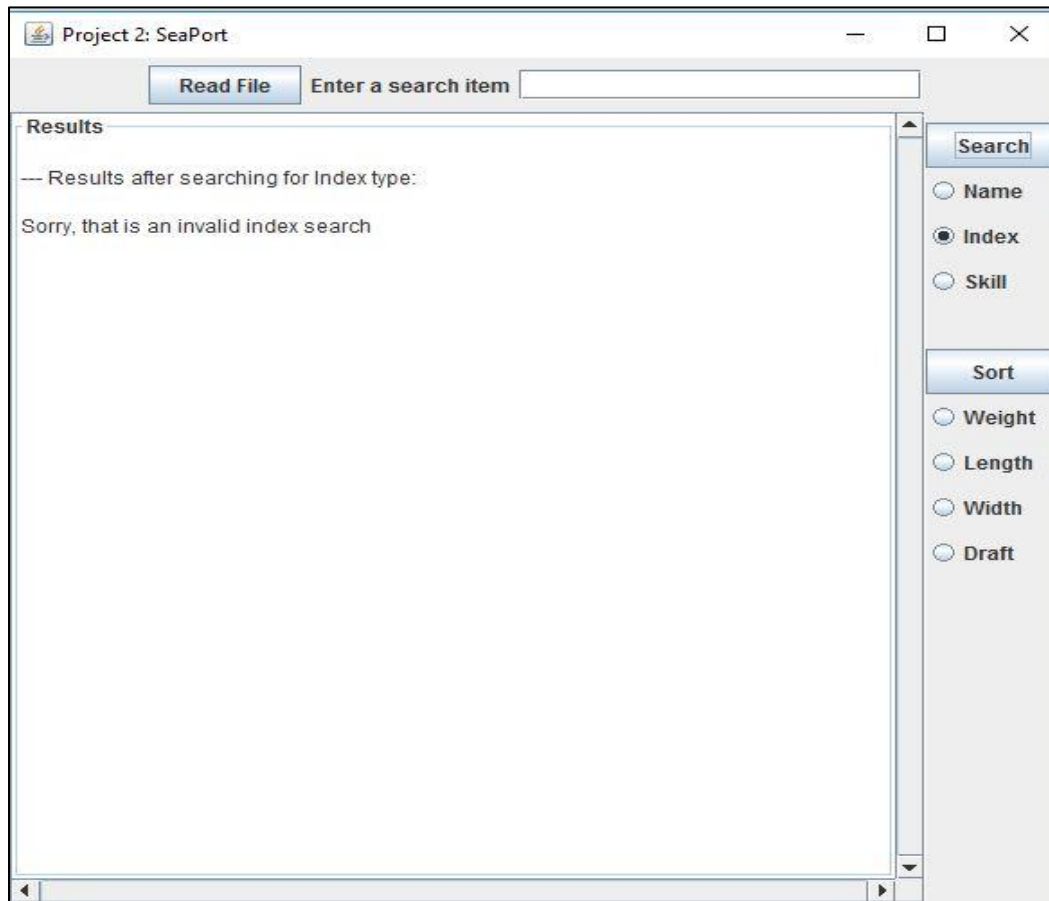
Search

☐ Name
☐ Index
☒ Skill

Sort

☐ Weight
☐ Length
☐ Width
☐ Draft

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

Project 2: SeaPort

Read File Enter a search item

Search

- ☐ Name
- ☐ Index
- ☐ Skill

Sort

- ☒ Weight
- ☐ Length
- ☐ Width
- ☐ Draft

Cargo ship: Bloom	40002
Weight:	79.9
Passenger ship: Venus	30004
Weight:	86.74
Cargo ship: Chestnut	40000
Weight:	120.85
Passenger ship: Mersey	30000
Weight:	125.99
Passenger ship: Newark	30001
Weight:	126.38
Passenger ship: Cromarty	30002
Weight:	134.41
Passenger ship: Ford	30003
Weight:	149.85
Cargo ship: Dane	40004
Weight:	189.12
Cargo ship: Joy	40001
Weight:	200.8
Cargo ship: Jetty	40003
Weight:	219.92

Sort by Ship Length: Enter in length

Project 2: SeaPort

Read File Enter a search item length

Passenger ship: Cromarty 30002
Length: 156.96

Cargo ship: Bloom 40002
Length: 234.26

Passenger ship: Mersey 30000
Length: 234.7

Cargo ship: Joy 40001
Length: 242.33

Passenger ship: Newark 30001
Length: 358.27

Cargo ship: Chestnut 40000
Length: 362.55

Cargo ship: Jetty 40003
Length: 443.54

Cargo ship: Dane 40004
Length: 448.99

Passenger ship: Venus 30004
Length: 450.43

Passenger ship: Ford 30003
Length: 483.92

Search

☐ Name

☐ Index

☐ Skill

Sort

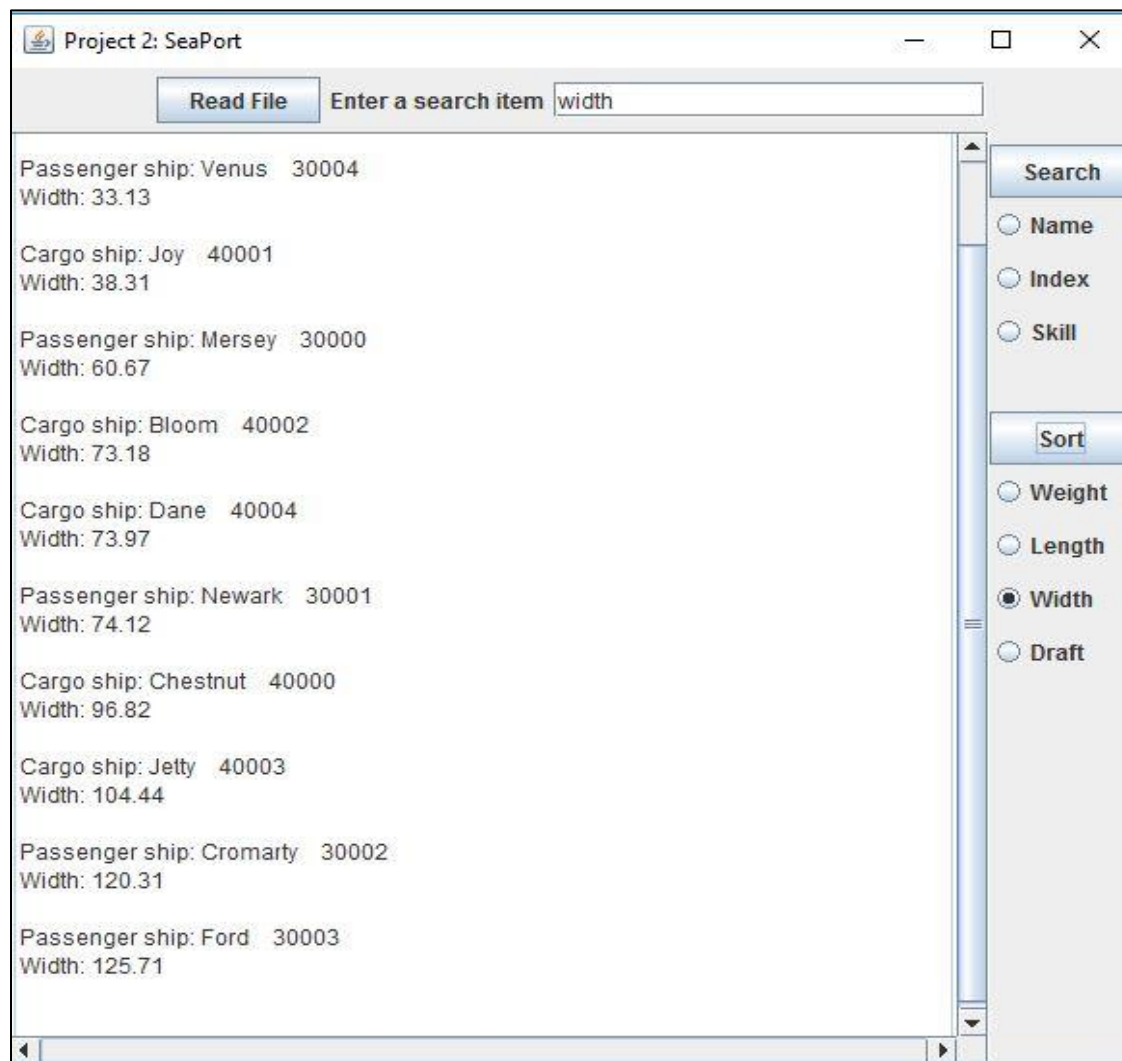
☐ Weight

☒ Length

☐ Width

☐ Draft

Sort by Ship Width: Enter in width



Project 2: SeaPort

Read File Enter a search item width

Search

☐ Name

☐ Index

☐ Skill

Sort

☐ Weight

☐ Length

☒ Width

☐ Draft

Passenger ship: Venus	30004	Width: 33.13
Cargo ship: Joy	40001	Width: 38.31
Passenger ship: Mersey	30000	Width: 60.67
Cargo ship: Bloom	40002	Width: 73.18
Cargo ship: Dane	40004	Width: 73.97
Passenger ship: Newark	30001	Width: 74.12
Cargo ship: Chestnut	40000	Width: 96.82
Cargo ship: Jetty	40003	Width: 104.44
Passenger ship: Cromarty	30002	Width: 120.31
Passenger ship: Ford	30003	Width: 125.71

Sort by Ship Draft: Enter in draft

Project 2: SeaPort

Read File Enter a search item draft

Cargo ship: Bloom 40002	
Draft: 15.71	
Cargo ship: Chestnut 40000	
Draft: 19.09	
Cargo ship: Joy 40001	
Draft: 23.49	
Passenger ship: Ford 30003	
Draft: 31.21	
Passenger ship: Newark 30001	
Draft: 31.54	
Cargo ship: Jetty 40003	
Draft: 34.16	
Passenger ship: Cromarty 30002	
Draft: 35.2	
Passenger ship: Mersey 30000	
Draft: 37.14	
Cargo ship: Dane 40004	
Draft: 37.67	
Passenger ship: Venus 30004	
Draft: 41.67	

Search

☐ Name

☐ Index

☐ Skill

Sort

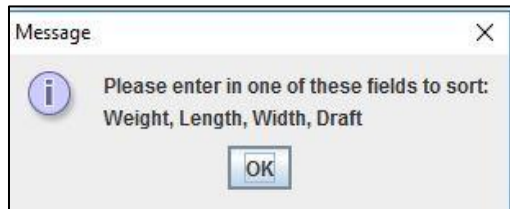
☐ Weight

☐ Length

☐ Width

☒ Draft

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



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

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

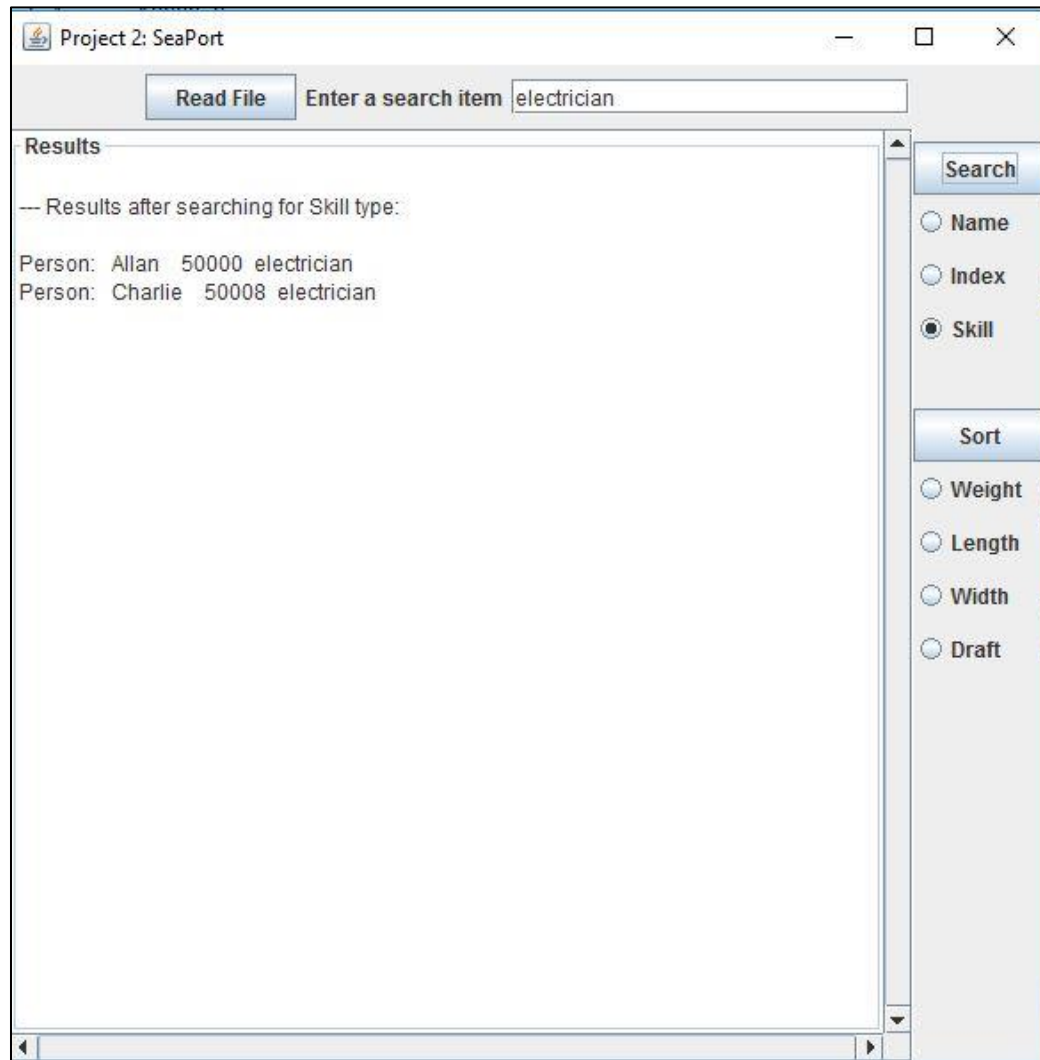
- **Special Features:**

- 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)

- **Screen Shots:**

- See screenshots of the program below for duplicates of both searching and sorting

Example of Searching and Returning Duplicates: (Ex: searching electrician and selecting skill)



Example of Sorting and Returning Duplicates: (Ex: sorting the ships weight and selecting weight)

Project 2: SeaPort

Read File Enter a search item weight

Results

- Cargo ship: Conscious 40002
Weight: 79.9
- Passenger ship: Aaft 30004
Weight: 86.74
- Cargo ship: Brock 40000
Weight: 120.85
- Passenger ship: Aardwolf 30000
Weight: 125.99
- Passenger ship: Ab 30004
Weight: 126.38
- Passenger ship: Broaden 30002
Weight: 134.41
- Passenger ship: Abaser 30003
Weight: 149.85
- Cargo ship: Dispraise 40004
Weight: 189.12
- Cargo ship: Brock 40001
Weight: 200.8
- Cargo ship: Consecration 40003
Weight: 219.92

Search

- ☐ Name
- ☐ Index
- ☒ Skill

Sort

- ☒ Weight
- ☐ Length
- ☐ Width
- ☐ Draft

III. **Test Plan:** (Update from Project 1)

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

Project 2 was an extension off project 1. The first thing I did before extending on to project 1 was to make sure that I understood what the instructions were asking for. The goal of this assignment was to implement a HashMap around the methods that we created in project 1. This required me to research the best way to implement a HashMap since it was the first time I ever did one. I found Oracles documentation as the best resource for understanding HashMap's. The next thing I did was create a compare method that will sort the ships by weight, length, width, and draft. I then added a sort function in the World class to implement this compare method. Finally, if we go to the UI class SeaPortProgram, I added radio buttons for weight, length, width, and draft. The sorting is done using the sort method in the World class, as well as, the compareTo switch statement in the Ship class. I knew that if I did this I would get the results I want. The program is now able to sort by these fields and display them nicely on the GUI output display.

IV. **Lessons Learned:** (Update from Project 1)

After completing project 2 there are many additional things that were learned from project 1. Like in project 1, I realize how important it is to work with multiple classes using inheritance and polymorphism. This project taught me how to use HashMap's, and compare methods to sort elements in a file. As you can see, the program is now able to sort the ships weight, length, width, and draft in ascending order. Like in previous programming classes, and in project 1, understanding what the problem is asking, and how to implement a solution to solve the problem only comes through trial and error. Project 2 involved a lot of trial and error. I would have to say the hardest part of this project was implementing the HashMap to support efficient linking of the classes that were used in project 1. What made this difficult was getting it to work with the readFile method, or in my case the process method, which is in the World class. After I could get this to work, implementing the comparable interface, which compares values and then returns an int which tells if the values compare less than, equal, or greater than

was less difficult since I have used this in previous programming classes. All in all, like in project 1, I spent a total of 35 hours on it this week, and factored in about 5 hours a day, or more.