Project 2

Nabeel Hussain

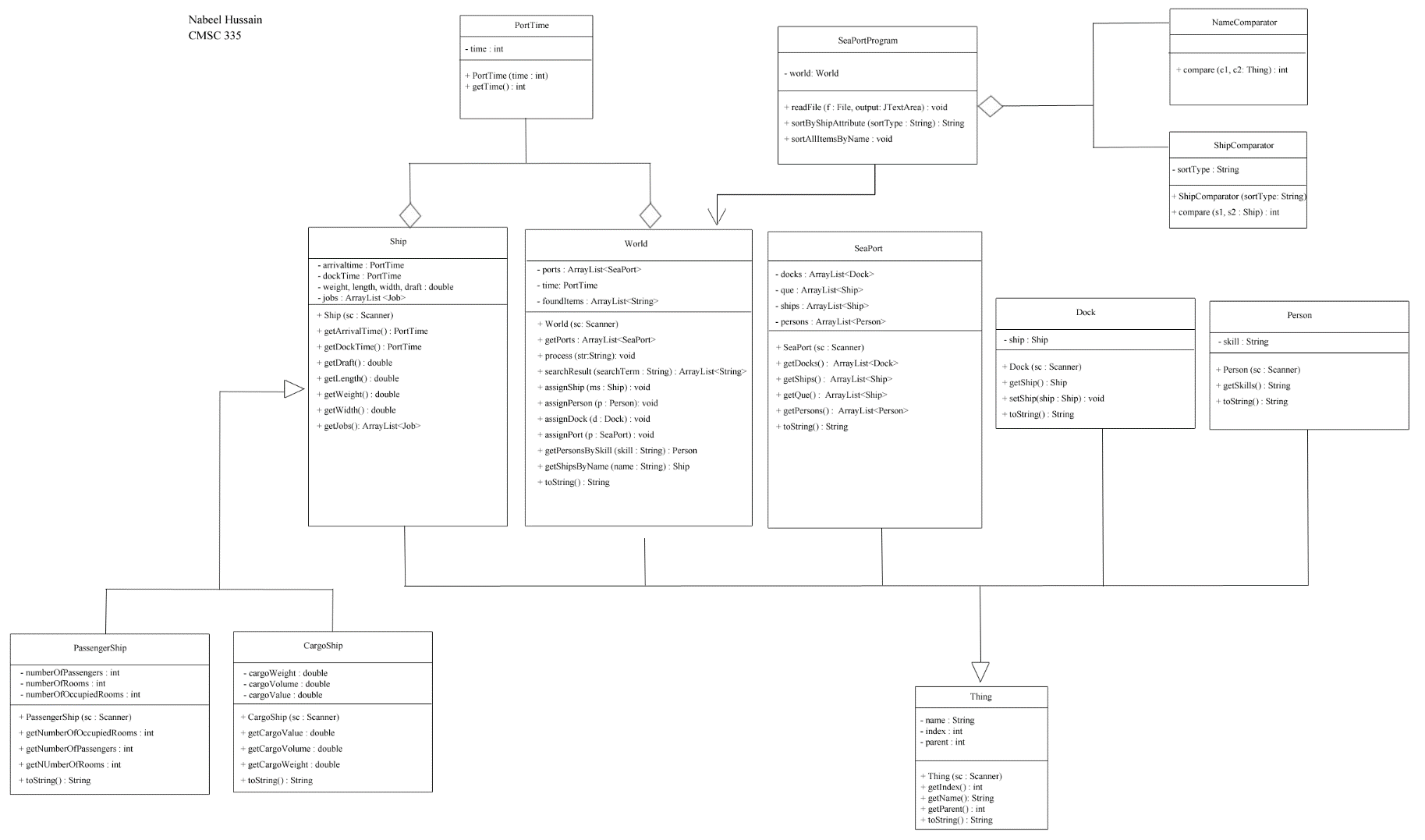
University of Maryland, University College

CMSC 335

September 14th, 2017

­

**Design**

UML Diagram:

*\** My UML Diagram is too large to fit within the width of this word document, so you will need to zoom in to see it. I have also attached a .png image of the UML as well in my submission so you will be able to more clearly see it there. \*

Project 2 involves the use of the same 11 classes that were used in Project 1, with the addition of two new compare classes. These first new class I created is called ShipComparator, which is used for sorting only the ship data by weight, length, width, or draft. The second new class I created is called NameComparator, which is used for sorting all the items by name.

As with project 1, the class named **Thing** is the parent class of an item, which holds the name, index, and parent data. The **World**, **SeaPort**, **Dock,** **Ship**, **Person**, and **Job** classes all extend the class Thing, meaning they inherit all the features of it, and they also have their own additional features to distinguish them from one another. The class **PassengerShip** and **CargoShip** are both subclasses of the class Ship, as they hold their own additional data to distinguish what a passenger ship and cargo ship are.

The class **World** is where the data from the text file will be read, processed, and stored into internal data structures. In this class, we want to read the text file line by line and store the pieces of data from each valid line into the correct data structures, using the Scanner class. The **SeaPortProgram** holds the GUI by extending the JFrame class, and it will be used to display the contents of the project. This class uses an instance of the class World to make the program functional and retrieve the internal data structure to be displayed. There is also a class called **PortTime**, which was not used for project 1 but it will be used to keep track of ship port and dock times.

All these classes have their own appropriate toString() method so that the data can be properly formatted for display in the gui. Also, all the necessary classes have an appropriate Scanner constructor, allowing the class to take advantage of super constructors, and any particular constructor focusing only on the addition elements of interest to that particular class.

More additional info on each class’s methods and variables can be found in the source code comments of each class.

**User’s Guide**

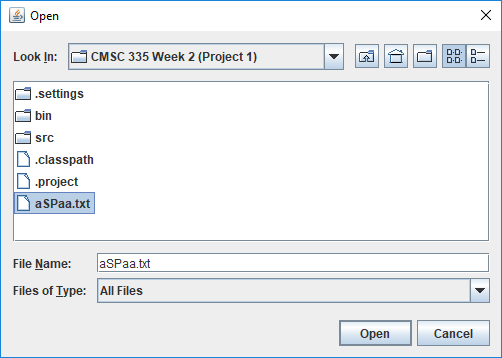
Given all the necessary java files and the correct .txt file for input, a user would be able to run my project in an IDE such as Eclipse. They would need to have all the class java files in the same package or folder and use the class called SeaPortProgram to run the project. Once the “Run” button is pressed in Eclipse, a JFileChooser popup window will appear asking the user to select a file for input. For my project, the user will select the **asPaa.txt** sample test file, and once it’s opened, the program will display a GUI that displays the formatted contents of the file in a text area with a scroll pane.

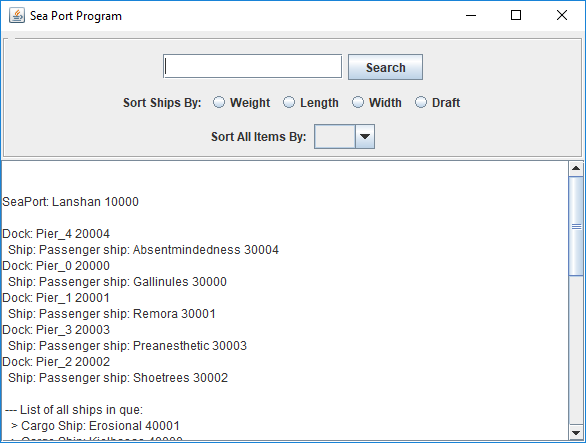
The GUI will also have a search field option, where the user can enter a search term by name, index, or skill, and it will display any matching results.

**In addition**, the gui will now incorporate sorting, where the user can sort the ships from the data file by weight, length, width, and draft. These sorting options will be displayed using radio button the user can select.

There will also be another option for the user to sort all the items by Name. They will be able to select this option by selecting the “Name” option from the JComboBox located underneath the radio buttons.

**Below is a screenshot of the GUI once the program has run**:

­­­

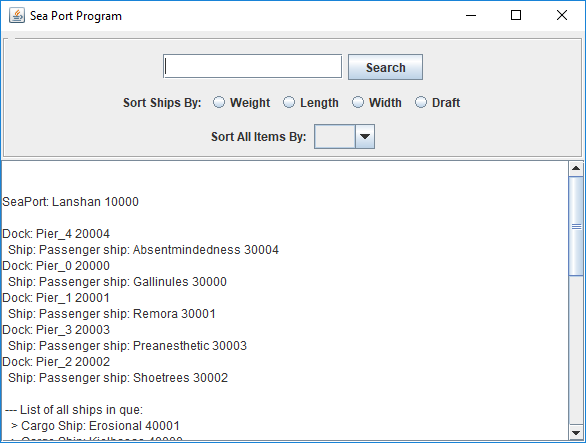


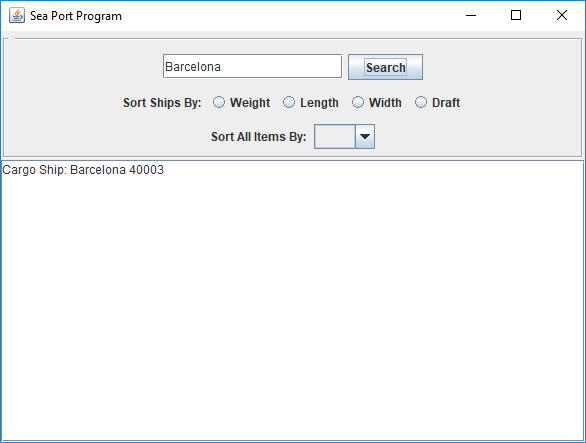
**Test Plan**

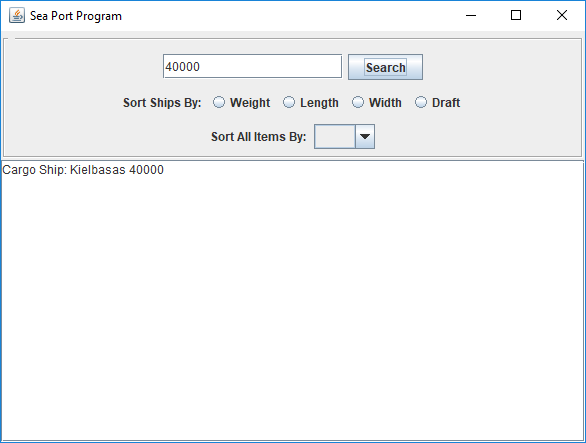
**Using input from aSPaa.txt**

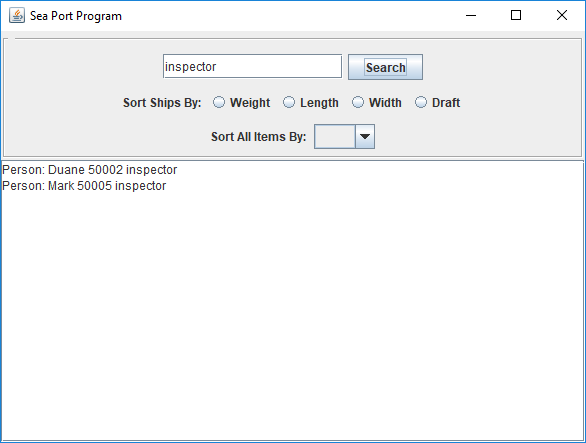
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input: aSPaa.txt** | **Expected Output** | **Did Test Pass?** |
| 1 | // port name index parent(null)  // port <string> <int> <int>  **port Lanshan 10000 0** | SeaPort: Lanshan 10000 | Y |
| 2 | // dock name index parent(port)  // dock <string> <int> <int>  **dock Pier\_4 20004 10000 30004** | Dock: Pier\_4 20004  Ship: Passenger ship: Absentmindedness 30004 | Y |
| 3 | // pship name index parent(dock/port) weight length width draft numPassengers numRooms numOccupied  // pship <string> <int> <int> <double> <double> <double> <double> <int> <int> <int>  **pship Gallinules 30000** **20000 125.99 234.70 60.67 37.14 746 246 246** | Passenger ship: Gallinules 30000 | Y |
| 4 | // cship name index parent(dock/port) weight length width draft cargoWeight cargoVolume cargoValue  // cship <string> <int> <int> <double> <double> <double> <double> <double> <double> <double>  **cship Erosional 40001 10000 200.80 242.33 38.31 23.49 172.73 188.54 235.57** | Cargo Ship: Erosional 40001 | Y |
| 5 | // person name index parent skill  // person <string> <int> <int> <string>  **person Sara 50000 10000 electrician** | Person: Sara 50000 electrician | Y |
| 6 | Search Target by Name  Search: **Barcelona** | Cargo Ship: Barcelona 40003 | Y |
| 7 | Search Target by Index  Search: **40000** | Cargo Ship: Kielbasas 40000 | Y |
| 8 | Search Target by Skill – Will return multiple items  Search: **inspector** | Person: Duane 50002 inspector  Person: Mark 50005 inspector | Y |
| 9 | Sort ship by Weight: smallest to largest  - Select “**Weight**” radio button | Cargo Ship: Generics 40002  Passenger ship: Absentmindedness 30004  Cargo Ship: Kielbasas 40000  Passenger ship: Gallinules 30000  Passenger ship: Remora 30001  Passenger ship: Shoetrees 30002  Passenger ship: Preanesthetic 30003  Cargo Ship: Toluene 40004  Cargo Ship: Erosional 40001  Cargo Ship: Barcelona 40003 | Y |
| 10 | Sort ship by Length: smallest to largest  - Select “**Length**” radio button | Passenger ship: Shoetrees 30002  Cargo Ship: Generics 40002  Passenger ship: Gallinules 30000  Cargo Ship: Erosional 40001  Passenger ship: Remora 30001  Cargo Ship: Kielbasas 40000  Cargo Ship: Barcelona 40003  Cargo Ship: Toluene 40004  Passenger ship: Absentmindedness 30004  Passenger ship: Preanesthetic 30003 | Y |
| 11 | Sort ship by Width: smallest to largest  - Select “**Width**” radio button | Passenger ship: Absentmindedness 30004  Cargo Ship: Erosional 40001  Passenger ship: Gallinules 30000  Cargo Ship: Generics 40002  Cargo Ship: Toluene 40004  Passenger ship: Remora 30001  Cargo Ship: Kielbasas 40000  Cargo Ship: Barcelona 40003  Passenger ship: Shoetrees 30002  Passenger ship: Preanesthetic 30003 | Y |
| 12 | Sort ship by Draft: smallest to largest  - Select “**Draf**t” radio button | Cargo Ship: Generics 40002  Cargo Ship: Kielbasas 40000  Cargo Ship: Erosional 40001  Passenger ship: Preanesthetic 30003  Passenger ship: Remora 30001  Cargo Ship: Barcelona 40003  Passenger ship: Shoetrees 30002  Passenger ship: Gallinules 30000  Cargo Ship: Toluene 40004  Passenger ship: Absentmindedness 30004 | Y |
| 13 | Sort all items by Name in alphabetical order: A to Z  - Select “**Name**” from JComboBox selections | SeaPort: Lanshan 10000  Dock: Pier\_0 20000  Ship: Passenger ship: Gallinules 30000  Dock: Pier\_1 20001  Ship: Passenger ship: Remora 30001  Dock: Pier\_2 20002  Ship: Passenger ship: Shoetrees 30002  Dock: Pier\_3 20003  Ship: Passenger ship: Preanesthetic 30003  Dock: Pier\_4 20004  Ship: Passenger ship: Absentmindedness 30004  --- List of all ships in que:  > Cargo Ship: Barcelona 40003  > Cargo Ship: Erosional 40001  > Cargo Ship: Generics 40002  > Cargo Ship: Kielbasas 40000  > Cargo Ship: Toluene 40004  --- List of all ships:  > Passenger ship: Absentmindedness 30004  > Cargo Ship: Barcelona 40003  > Cargo Ship: Erosional 40001  > Passenger ship: Gallinules 30000  > Cargo Ship: Generics 40002  > Cargo Ship: Kielbasas 40000  > Passenger ship: Preanesthetic 30003  > Passenger ship: Remora 30001  > Passenger ship: Shoetrees 30002  > Cargo Ship: Toluene 40004  --- List of all persons:  > Person: Archie 50003 captain  > Person: Betsy 50004 cleaner  > Person: Duane 50002 inspector  > Person: Duane 50006 clerk  > Person: Mark 50005 inspector  > Person: Sara 50000 electrician  > Person: Thomas 50001 clerk | Y |

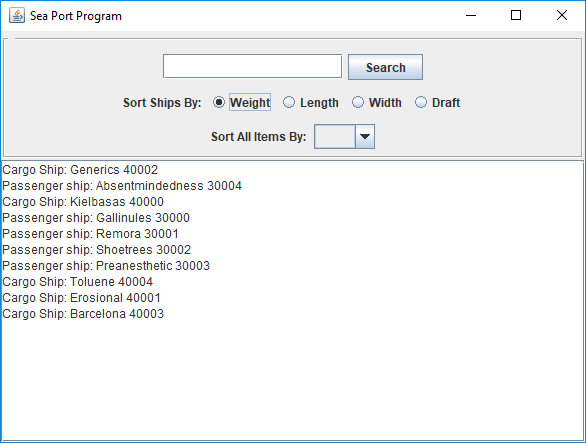
**Some Test Case Screenshots:**

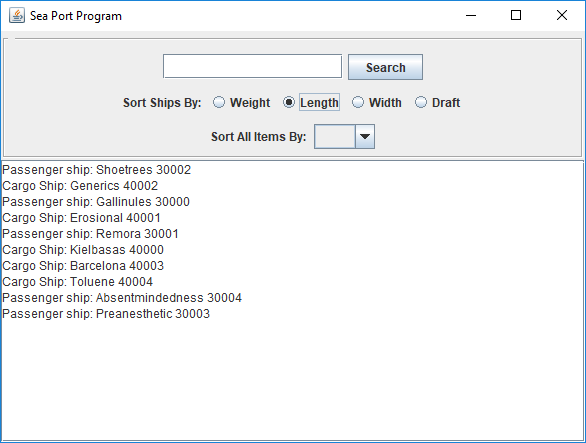


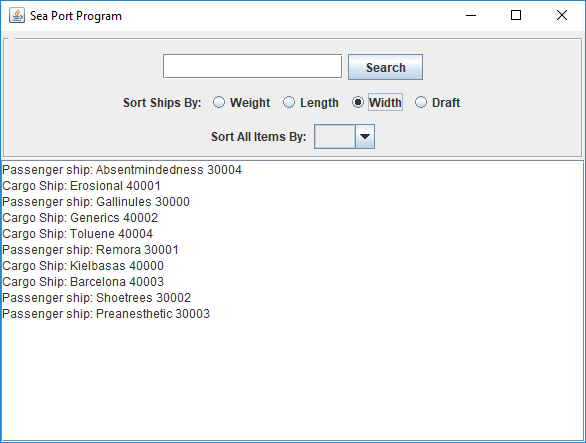


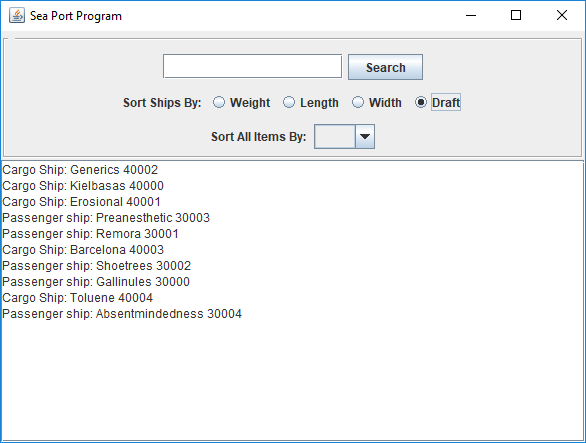


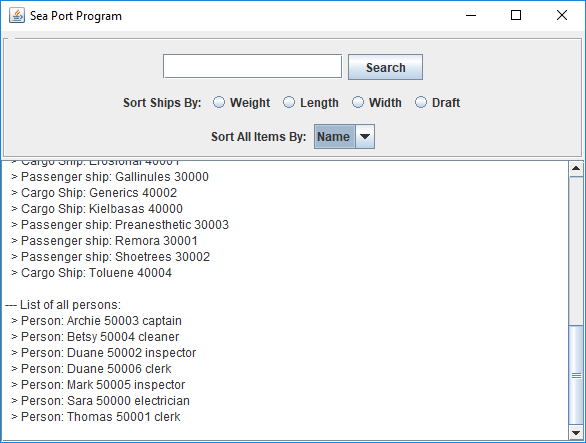












**Lessons Learned**

While working on Project 2, there were a lot of new things I was able to learn and get experience with in terms of completing this project. It was a bit challenging for me at first to understand what needed to be done to improve the code from Project 1, but after getting a few questions answered, I was able to complete it in a reasonable time. The use of HashMaps to improve the efficiency of searching for items was new to me, but after I was able to read up on it some more, I found it easy to understand.

I created instances of the HashMap objects for each type of item that was going to be read in from the data file in my World class. This allowed me to update all the assign methods from project 1, such as assignShip, assignDock, assignPort, and assignPerson, so that they added the items to the hash map as well. Also, by using the get method of the HashMap, I was able to retrieve the items by their index numbers, allowing me to eliminate the getDockByIndex, getShipByIndex, getPortByIndex, and gerPersonByIndex methods I used in Project 1. The purpose of this was to change the operation of searching for an item with a particular index from an O(N) operation, i.e. searching through the entire data structure to see if the code can find the parent index parameter, to an O(1) operation, a hash map lookup.

The use of the Comparator class to help with sorting all the items by name and the ships by weight, length, width, and draft was more of a review for me. I had used the Comparator class in previous classes, so I was able to understand the sorting part of this assignment much easier than the HashMaps part.

Overall, this assignment was challenging, however it helped me understand and reinforce all the material I learned from this week’s readings. I hope to take the knowledge I obtained from this project and apply it to future projects as well.