Project 1

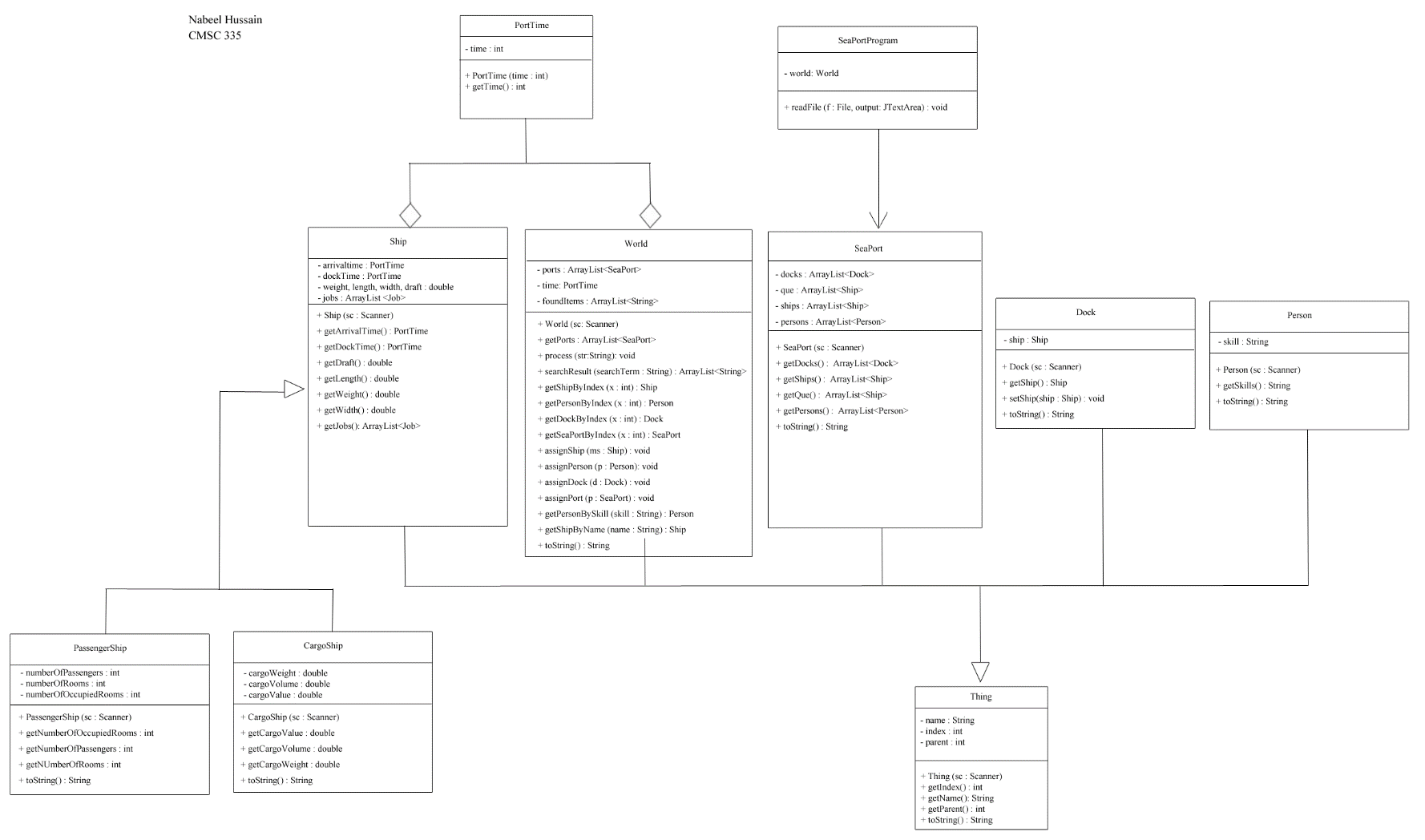
Nabeel Hussain

University of Maryland, University College

CMSC 335

September 2nd, 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 1 involves the use of 11 classes, but there is one class named **Job**, which will not be used until Projects 3 and 4 per the instructions. 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.

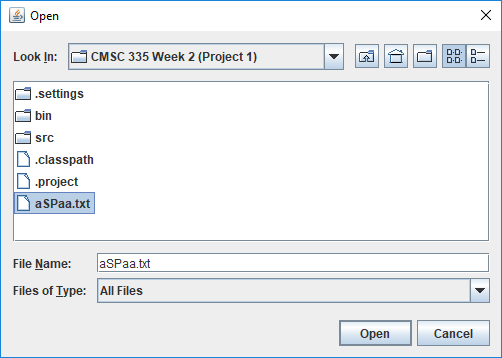
More additional info on each class’s methods and variables can be found in the source code comments of each class.

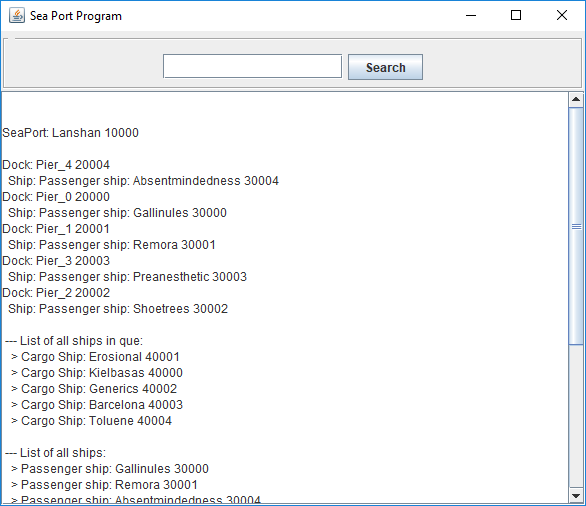
**User’s Guide**

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

**Below are some screenshots of the program once it has run**:

­­­

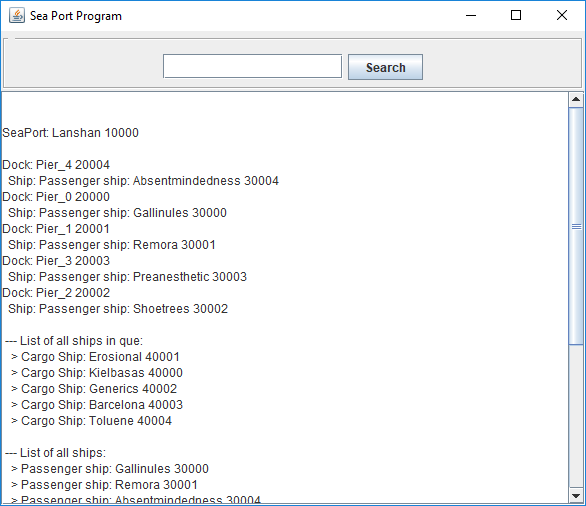


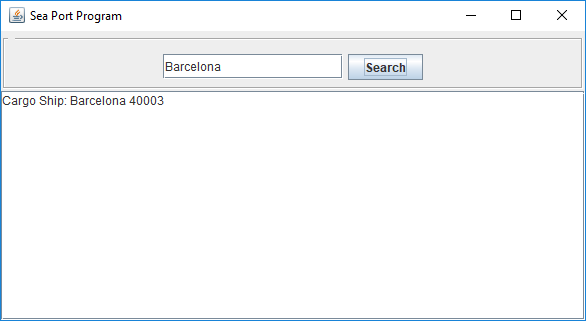
**Test Plan**

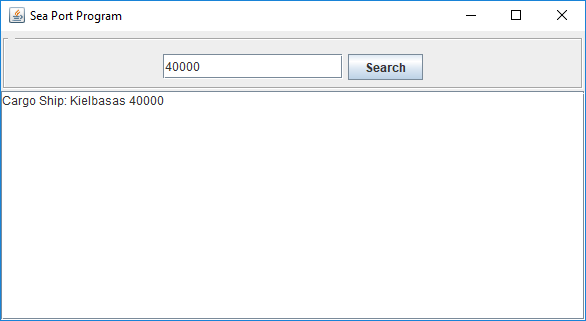
**Using input from aSPaa.txt**

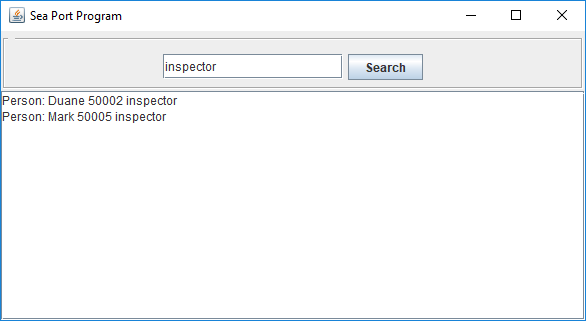
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input: aSPaa.txt** | **Expected Output** | **Did Test Pass?** |
| 1 | // port name index parent(null)  // port <string> <int> <int>  **port Lanshan 10000 0** | SeaPort: Lanshan 10000 | Y |
| 2 | // dock name index parent(port)  // dock <string> <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 |

**Some Test Case Screenshots:**









**Lessons Learned**

While working on Project 1, there were a lot of familiar concepts that I was able to refresh my memory on from previous programming classes. The concepts of parent-child relationships, inheritance, polymorphism, and encapsulation were all incorporated in one way or another in this project, which helped me to further enhance my understanding of these topics. It was a bit challenging for me at first to understand this project with all the numerous classes needing to be incorporated, but as I re-read the instructions multiple times I was able to break everything down into manageable sections and work my way through each part.

Due to most of the concepts being all review to me from previous class projects, there was not really anything new that I learned. However, it was a good project to refresh my memory on the things I had forgotten.

The most difficult part of this assignment in my mind was figuring out how to find and display the correct results of a search. In the instructions we were given a sample getShipByIndex(int x) method, and I used that as a model for creating the methods for other key search targets. For example, if searching for name, I created similar methods for port, dock, person, and ship, such as getShipByName(String name), etc.... Also, when searching for a skill, I created a method called getPersonsBySkill(String skill). I used these methods to find the correct results when searching by name, index, or skill. Each time a match was found, I added the item into a String ArrayList called **foundItems**, to keep track of those searches that returned multiple items.

Overall, this assignment was moderately challenging, and 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.