Seaport Project: Part #1

Nicholas Mills

CMSC 335: Object Oriented and Concurrent Programming

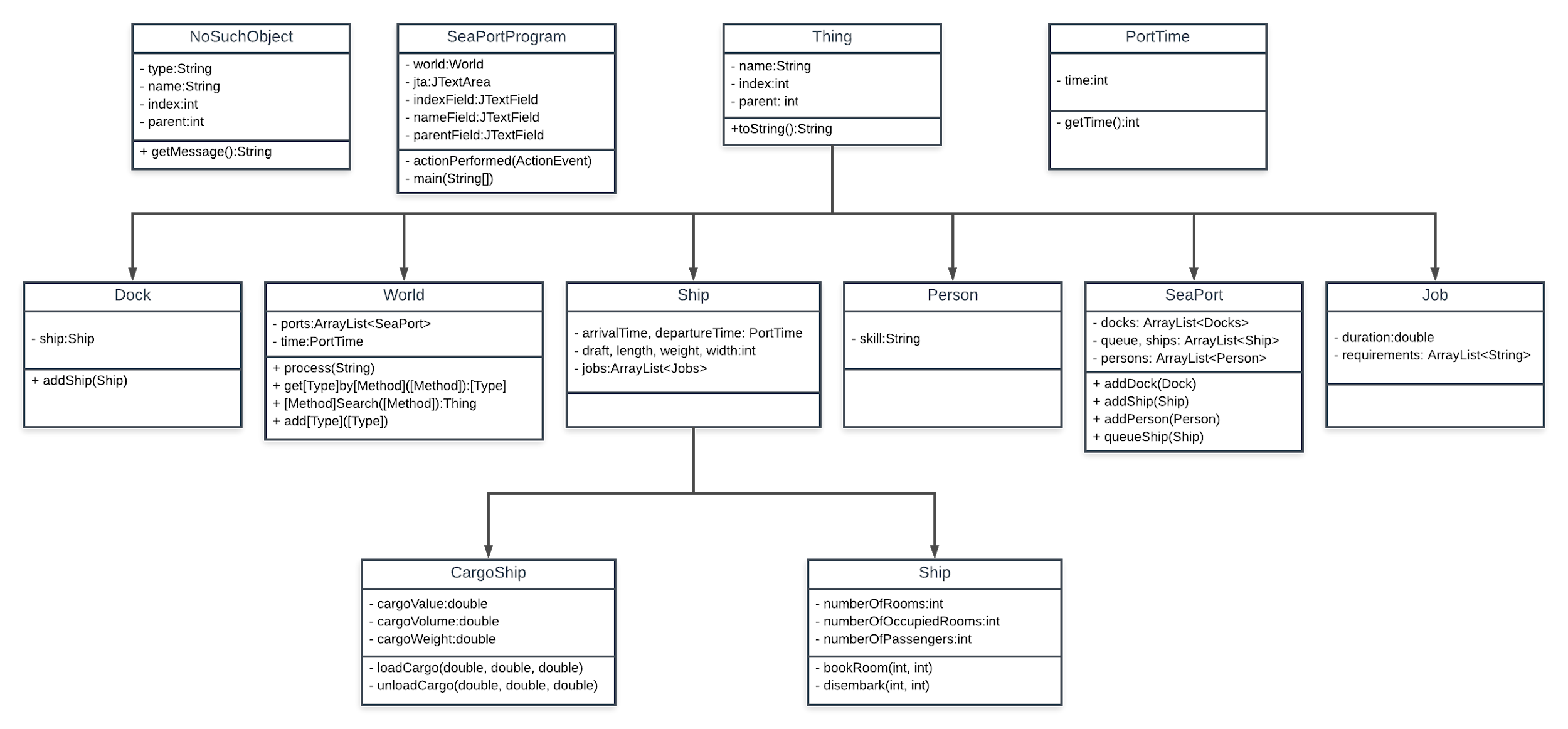
University of Maryland University College

Mihaela Dinsoreanu

11/3/2018

Design

**UML Diagram**



**Class Description**

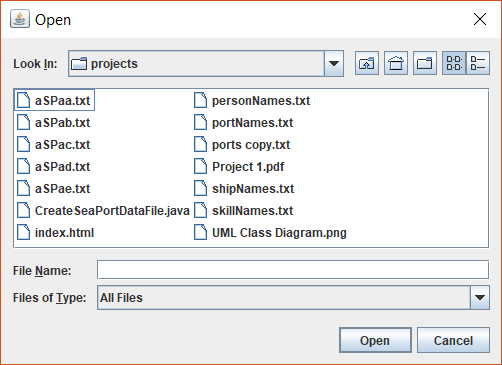
This portion of the SeaPort project led me to mainly focus around the SeaPortProgram and World classes due to the heavy focus on building the World and GUI, which is contained in SeaPortProgram. I also created a custom exception called NoSuchObject to handle inconsistencies within source files and unsuccessful searches.

**SeaPortProgram.** In this class, I allowed the user to select the file of their choice using a JFileChooser that starts off in the project directory, as requested by the project, as well as displaying the output of the file in a JTextArea in the center of the window. I also included error handling for if the source file wasn’t found and also detection for if there are some problems with the source file. I then created a subpanel for the search bars that used a 3x3 grid, each row containing a label for the type of search that is being done, a text field for inputting the search terms, and a button to search the world for those terms. The results of these searches are printed in the JTextArea in the middle. If there is no result, a dialog box will appear informing the user. I also included error handling for incorrect formats for the numeric fields. Below the search bars, there is a reset button to return the JTextArea to its original state of displaying the entire output of the source file.

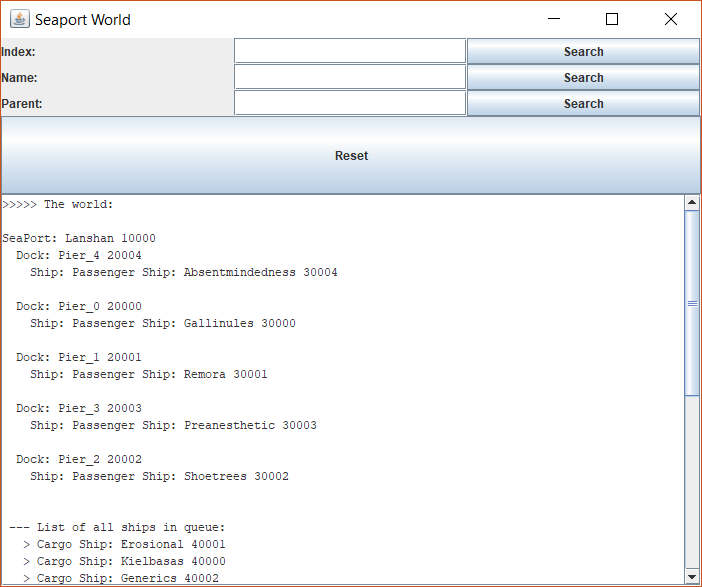
**World.** This class contains logic for building and searching the world. The constructor takes a scanner of the source file and passes information from it one line at a time to the process method. This method then determines the correct type of Thing and attaches it to the appropriate parent Thing. Search methods are also included for each of the types of Thing, as well as a nonspecific search method which will check for all types of Thing that match the search term.

**NoSuchObject.** This exception is thrown whenever a search method comes up with nothing. It captures the type of search occurring and generates an appropriate error message through the getMessage method.

User’s Guide

When SpacePortProgram is run, a dialog box will appear. Select a source file from the list contained in the ./src/projects subfolder. Valid files start with an a and are .txt files. 

You can also make your own source files using CreateSeaPortDataFile.java. Press open. This will display the following window:



A text output of the world can be found in the text box in the lower half of the window. You can use the scroll bar to look through the data. From here, you can search the world by index, name, or parent. Type what you wish to search into the appropriate field and press the accompanying search button. The results will appear in the text box, replacing the existing information. To restore the window to its original state, press reset.

Test Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Purpose: | Input: | Expected Output: | Actual Output: | P/F |
| Base Test | aSPaa.txt | Window with formatted output |  | P |
| Index Search | aSPaa.txt  20004 | Pier\_4 toString() |  | P |
| Name Search | aSPaa.txt  Shoetrees | Shoetrees toString() |  | P |
| Parent Search | aSPaa.txt  10000 | Children of Lanshan toString() |  | P |
| Reset | aSPaa.txt | Same as Test Case #1 |  | P |
| Bigger File | aSPab.txt | Window with formatted output |  | P |
| NoSuchObject index | aSPab.txt  10 | The Object 10 was not found. |  | P |
| NoSuchObject name | aSPab.txt  blargh | The Object with the name of blargh was not found. |  | P |
| NoSuchObject parent | aSPab.txt  10 | The Object with the parent of 10 was not found. |  | P |

Comments

I realized after I had finished that the search options would be better suited for radio buttons. I will change that in the next iteration of the project. Additionally, I will have to rework the NoSuchObject exception message generation if many more search options are added. There may also be a way to condense the code for the search methods but that will require more analysis.

Lessons Learned

Early on in the project, I was having trouble with a NullPointerException. It took me a bit to figure out where it was coming from. I realized eventually that I was trying to add something to an uninitialized list. This simply reiterates the importance of initializing variables. I also learned how to use JFileChoosers, since I had not done that previously. Lastly, I also learned how to use for-each loops, as I also have not used those in the past.