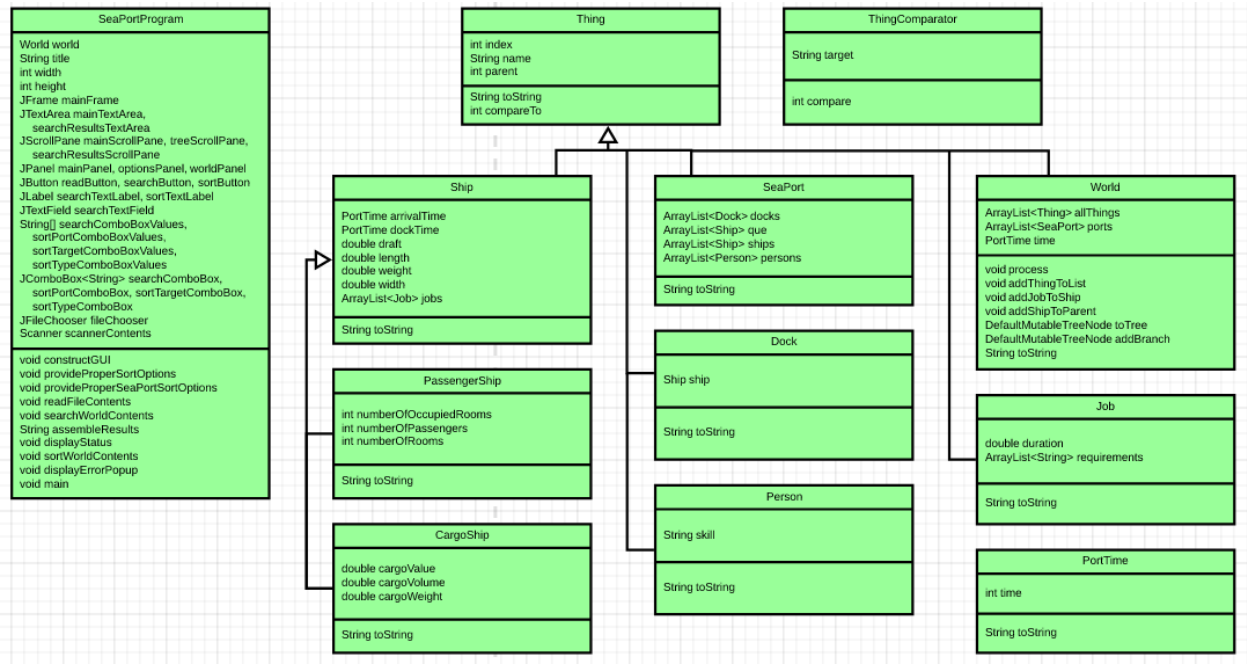
Project 2 Documentation

01/29/18

Andrew Eissen

UMUC CMSC 335

Part I: Design



(Author note: Getters/setters not included in UML diagram due to a lack of space and an inability to make the entire diagram fit in one frame to screen capture. Relatedly, all images, diagram and screenshots included, have been included in a zip file for convenient HD viewing.)

As per the Project 2 design rubric, the author assembled all classes as they appeared in the first project, augmented via the inclusion of the new `ThingComparator` class for use in sorting `Thing` objects. As per the first project, the only unused class was `PortTime` which was mentioned nowhere in the rubric and was simply included due its mention on the first page. Similarly, as stated in the Project 1 Documentation, the only private class field not on the rubric's listing of fields/methods is `allThings`, an `ArrayList` of `Things` contained within new `World` instances and used to store a quick and convenient listing of all extant `World` objects.

Part II: User's Guide

Users unfamiliar with the process of running Java programs may have some difficulty running Project 2 without some prior experience and a set of qualifications. Users must have Java installed on their computer and must have administrator privileges on their chosen machine. Pointing the Windows command prompt to the folder containing the files, users must first compile the classes by typing `javac SeaPortProgram.java`. Once compiled, users may enter `java SeaPortProgram` to run the program and begin interacting with the GUI. Alternatively, if users possess an IDE like NetBeans, they can open `SeaPortProgram.java` and run the program from within after creating a new package.

To prevent invalid input, the program will not open non-text files or ill-formatted text files, so users must make an effort to open a file that conforms to the organizational layout of the sample data files. It should be noted that the use of search bar necessitates the inclusion of proper input. Case and spelling are both integral to retrieving the desired results, as the program will not find results that are improperly formatted as they appear in the text file.

Part III: Test Plan

Prior to the start of the second project, the author sat down and considered the manner in which data and the UI could be refigured to best enhance usability. The decision to add a panel for a `JTree` representation of the `World` instance was considered after the author's reading of the Week 4 Java documentation material. Discovering the `JTree` and the manner in which it could be used to provide an interactive way of visualizing the world, the author went

about studying the specifics of how such a tree could be constructed from the data within the text file. The subsequent implementation was added to a left panel.

Furthermore, the author was dissatisfied with the manner in which search results were displayed, believing that large files with more data would result in unreasonably large `JOptionPane` sizes that would extend into unreadable territory. As such, the author replaced this method with a simple GUI panel that displays a log of all searches within a `JScrollPane` for maximum readability, appending new searches to the bottom of the `JTextArea`. The inspiration for this method was taken from a number of JavaScript userscripts the author has written in the past to automate certain tasks, the GUIs of which include an operation log at the bottom to provide the user with status updates based on the success/failure of the operations being performed.

Part III b: Test Cases

As with the previous project, the same two files were employed for the following tests. `aSPaa.txt` was retrieved from the `.zip` file provided on LEO, while `asPad.txt` was assembled via `CreateSeaPortDataFile.java`. Both files are of differing lengths, with former including only one port and no jobs and the latter including several ports and many jobs.

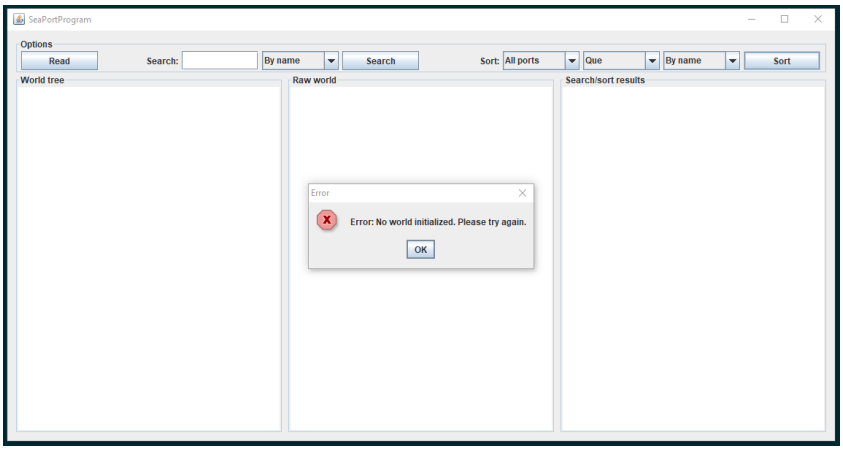
As the previous project tested cases related to the search function and program reaction to improper input, this documentation test table will include only test cases related to the specific functions required for Project 2. Details on the previous project may be retrieved in that project's documentation file as required.

Text File	Input	Expected Output	Output
N/a	Sort button	Error: No world initialized. Please try again.	Error: No world initialized. Please try again.
aSPaa.txt	All ports Jobs By name	Sort results for 'Jobs by name in all ports' > No results found.	Sort results for 'Jobs by name in all ports' > No results found.
aSPaa.txt	All ports Persons By name	Sort results for 'Persons by name in all ports' > Archie (50003) > Betsy (50004) > Duane (50002 > Sara (50000) > Thomas (50001)	Sort results for 'Persons by name in all ports' > Archie (50003) > Betsy (50004) > Duane (50002) > Sara (50000) > Thomas (50001)
aSPaa.txt	All ports Docks By name	Sort results for 'Persons by name in all ports' > Pier_0 (20000) > Pier_1 (20001) > Pier_2 (20002) > Pier_3 (20003) > Pier_4 (20004)	Sort results for 'Persons by name in all ports' > Pier_0 (20000) > Pier_1 (20001) > Pier_2 (20002) > Pier_3 (20003) > Pier_4 (20004)
aSPad.txt	Mexico_City Que By weight	Sort results for 'Que by weight in Mexico_City' > Reachers (58.57) > Foliages (76.78) > Ogler (90.5) > Hickory (109.21) > Baggages (113.31) > Disorientated (115.23) > Prolog (120.22) > Unmoored (122.53) > Cowboys (123.36) > Perpendicular (125.78) > Horsetail (127.36) > Fishlines (127.53) > Emasculate (133.78) > Fiancee (140.72) > Instincts (159.04) > Obnoxiously (166.56) > Cantonment (176.58) > Rumps (188.24) > Misstep (192.48) > Welcoming (205.78) > Highborn (206.62) > Ambassadorial (208.69) > Coterminous (216.28) > Recelebrated (224.0) > Encrust (225.6) > Incorrigible (228.59) > Forger (230.02) > Restuffs (237.94) > Saskatchewan (239.98) > Wrongful (244.83)	Sort results for 'Que by weight in Mexico_City' > Reachers (58.57) > Foliages (76.78) > Ogler (90.5) > Hickory (109.21) > Baggages (113.31) > Disorientated (115.23) > Prolog (120.22) > Unmoored (122.53) > Cowboys (123.36) > Perpendicular (125.78) > Horsetail (127.36) > Fishlines (127.53) > Emasculate (133.78) > Fiancee (140.72) > Instincts (159.04) > Obnoxiously (166.56) > Cantonment (176.58) > Rumps (188.24) > Misstep (192.48) > Welcoming (205.78) > Highborn (206.62) > Ambassadorial (208.69) > Coterminous (216.28) > Recelebrated (224.0) > Encrust (225.6) > Incorrigible (228.59) > Forger (230.02) > Restuffs (237.94) > Saskatchewan (239.98) > Wrongful (244.83)
aSPad.txt	Darkhan Que By draft	Sort results for 'Que by draft in Darkhan' > Confided (15.35) > Jellied (15.75) > Overate (16.95) > Strews (17.02) > Refracturing (18.52) > Ensue (18.89) > Integer (20.9) > Informality (24.03) > Oriented (24.14) > Potentiation (26.28) > Indefeasibly (26.32) > Vugs (27.02) > Aflame (28.15) > Monism (28.53) > Accidie (30.47)	Sort results for 'Que by draft in Darkhan' > Confided (15.35) > Jellied (15.75) > Overate (16.95) > Strews (17.02) > Refracturing (18.52) > Ensue (18.89) > Integer (20.9) > Informality (24.03) > Oriented (24.14) > Potentiation (26.28) > Indefeasibly (26.32) > Vugs (27.02) > Aflame (28.15) > Monism (28.53) > Accidie (30.47)

		> Liquidation (31.43) > Desiccating (33.56) > Reassumes (36.23) > Dragropes (36.84) > Chevrolets (37.22) > Clabber (37.61) > Diorama (38.4) > Cardamon (39.46) > Umbras (39.81) > Unsurely (40.04) > Quoted (42.48) > Accessorily (42.78)	> Liquidation (31.43) > Desiccating (33.56) > Reassumes (36.23) > Dragropes (36.84) > Chevrolets (37.22) > Clabber (37.61) > Diorama (38.4) > Cardamon (39.46) > Umbras (39.81) > Unsurely (40.04) > Quoted (42.48) > Accessorily (42.78)
aSPad.txt	All ports Persons By name	Sort results for 'Persons by name in all ports' > Aaron (50077) > Adrienne (50059) > Amber (50047) > Annie (50004) > Arthur (50061) > Austin (50030) > Becky (50069) > Bernard (50051) > Blanche (50031) > Bobbie (50049) > Brad (50020) > Brendan (50068) > Caleb (50066) > Candice (50036) > Charlene (50062) > Charlotte (50015) > Chris (50003) > Connie (50024) > Constance (50006) > Cory (50044) > Craig (50065) > Crystal (50053) > Dave (50064) > Donnie (50021) > Elsa (50081) > Enrique (50057) > Everett (50060) > Gabriel (50070) > Gayle (50007) > Grace (50073) > Gregg (50013) > Guy (50035) > Jack (50040) > Jack (50028) > Janis (50048) > Jay (50050) > Jean (50046) > Jermaine (50045) > Jessie (50032) > Jody (50026) > Joe (50023) > Joel (50000) > Jordan (50078) > Julia (50072) > June (50009) > Kayla (50019) > Kelvin (50054) > Ken (50042) > Laura (50010) > Laurence (50052) > Lonnie (50008) > Lorenzo (50018) > Lynn (50056)	Sort results for 'Persons by name in all ports' > Aaron (50077) > Adrienne (50059) > Amber (50047) > Annie (50004) > Arthur (50061) > Austin (50030) > Becky (50069) > Bernard (50051) > Blanche (50031) > Bobbie (50049) > Brad (50020) > Brendan (50068) > Caleb (50066) > Candice (50036) > Charlene (50062) > Charlotte (50015) > Chris (50003) > Connie (50024) > Constance (50006) > Cory (50044) > Craig (50065) > Crystal (50053) > Dave (50064) > Donnie (50021) > Elsa (50081) > Enrique (50057) > Everett (50060) > Gabriel (50070) > Gayle (50007) > Grace (50073) > Gregg (50013) > Guy (50035) > Jack (50040) > Jack (50028) > Janis (50048) > Jay (50050) > Jean (50046) > Jermaine (50045) > Jessie (50032) > Jody (50026) > Joe (50023) > Joel (50000) > Jordan (50078) > Julia (50072) > June (50009) > Kayla (50019) > Kelvin (50054) > Ken (50042) > Laura (50010) > Laurence (50052) > Lonnie (50008) > Lorenzo (50018) > Lynn (50056)

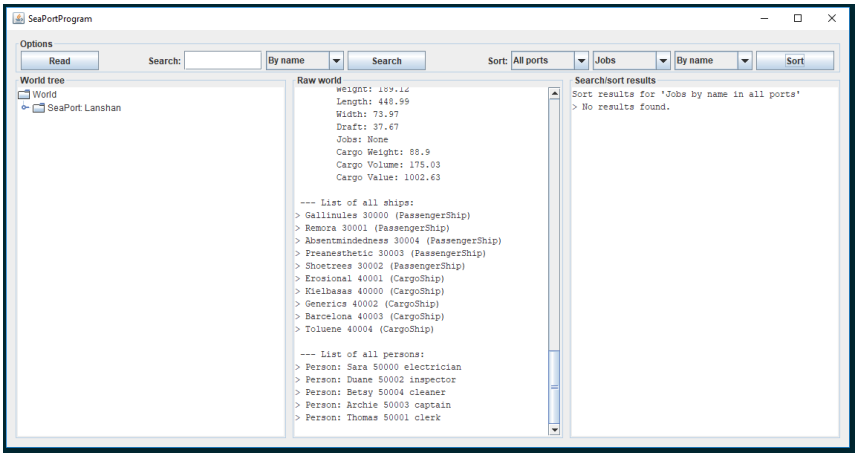
		> Lynn (50017) > Mack (50055) > Mark (50005) > Marsha (50034) > Matthew (50011) > Melanie (50079) > Micheal (50076) > Michele (50002) > Milton (50022) > Morris (50025) > Pamela (50075) > Patrick (50037) > Randy (50063) > Rex (50027) > Roger (50074) > Rolando (50041) > Ronnie (50058) > Rosemary (50033) > Ruth (50039) > Shannon (50001) > Shawna (50080) > Sheldon (50012) > Sonja (50043) > Stacy (50067) > Stacy (50029) > Tim (50016) > Tonya (50071) > Willie (50038) > Zachary (50014)	> Lynn (50017) > Mack (50055) > Mark (50005) > Marsha (50034) > Matthew (50011) > Melanie (50079) > Micheal (50076) > Michele (50002) > Milton (50022) > Morris (50025) > Pamela (50075) > Patrick (50037) > Randy (50063) > Rex (50027) > Roger (50074) > Rolando (50041) > Ronnie (50058) > Rosemary (50033) > Ruth (50039) > Shannon (50001) > Shawna (50080) > Sheldon (50012) > Sonja (50043) > Stacy (50067) > Stacy (50029) > Tim (50016) > Tonya (50071) > Willie (50038) > Zachary (50014)

Test Case 1



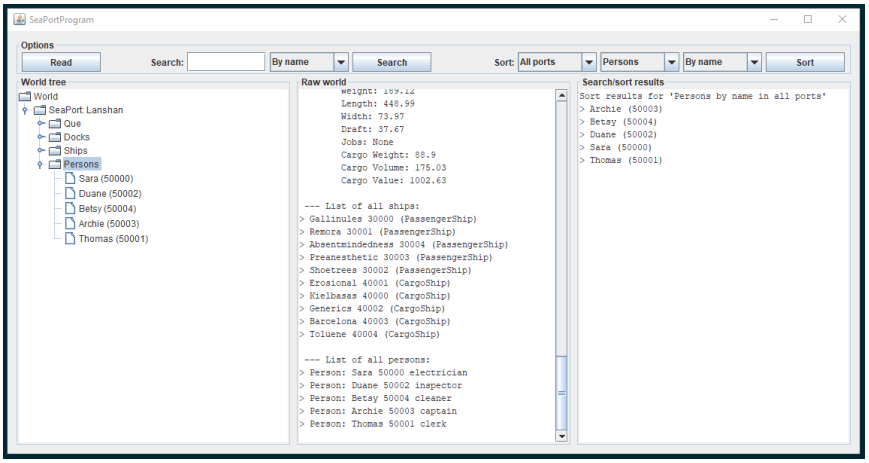
In the case of the first test example, the author tested the standard program-breaking test click of the “Sort” button prior to initialization of any text file or world. As expected, the program displays a warning modal forbidding such behavior.

Test Case 2



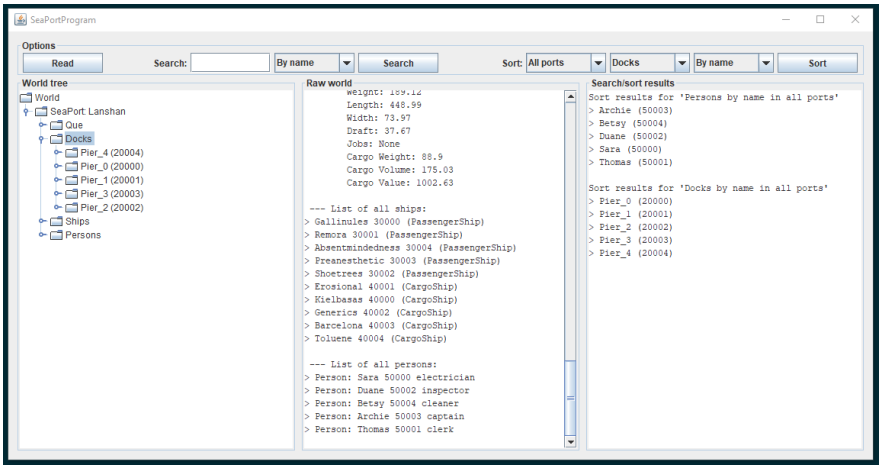
In the second test case, the author elected to sort the first data file for instances of `Jobs` aboard `Ships`. As the Project 2 rubric lists `Jobs` as optional until the third and fourth program, the file is bereft of `Jobs` and thus a “no results” message is displayed.

Test Case 3



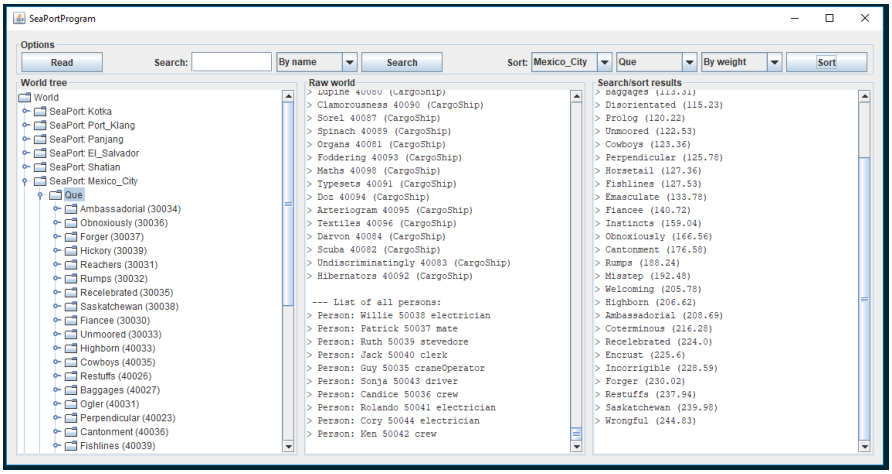
This test case tests the program’s response to standard legitimate sorting options, this time looking specifically for `Person` instances in the various `SeaPorts`. As expected, the appropriate `Things` are listed in order by name and appended to the search panel.

Test Case 4



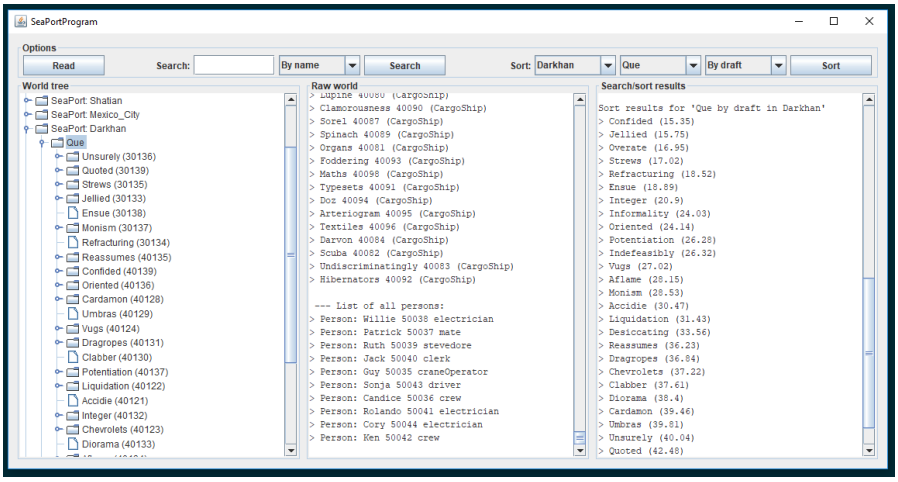
Immediately after the completion of the previous test case, the fourth test case was run to test the program's reaction to sorting searches of `Docks` by name. Much like the previous case, the `Docks` are displayed with their indices in parentheses.

Test Case 5



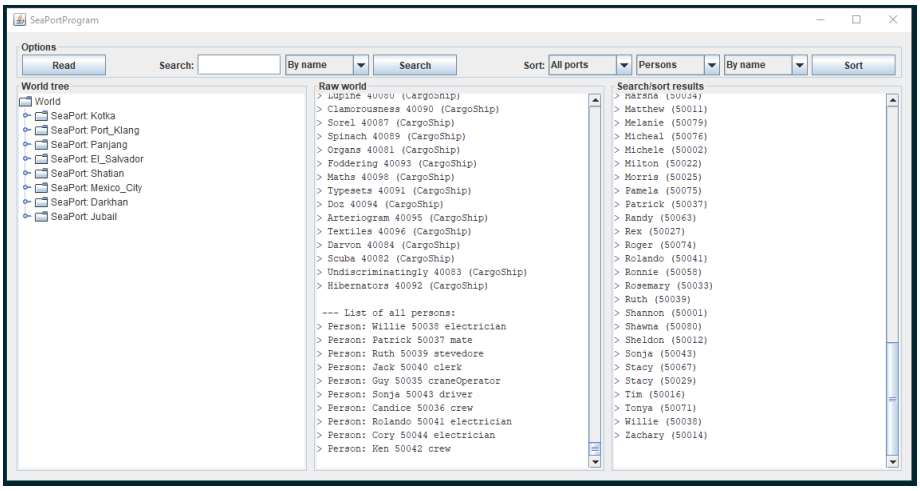
This test case handles large file data in the form of the second file, sorting through its contents in search of queue-bound `Ships` in Mexico City sorted by weight. As expected, the program lists a large number of instances, with the associated weight displayed to the right.

Test Case 6



The sixth test case employs the same text file as the previous, though in this case, queued vessels in Darkhan are sorted by draft. Once again, the corresponding draft value is displayed in parentheses beside each entry to aid in comparisons.

Test Case 7



As a test of the all `SeaPorts` option, the final test retrieves all `Person` instances in the world and sorts them by name, displaying all results as intended. This function was added by the author as a means of taking a look at the world and enabling global comparisons thereof.

Part IV: Comments

As far as future additions are concerned, the author has several areas of improvement highlighted for future consideration. First, the search methods could still do with some tweaking, given the fact that they still rely upon an iterative search through the entirety of all `World` objects in search of a search term matching that of one of their fields. This could and should probably be replaced by a set of `HashMaps` that list items by fields like name, index, etc. However, the author is still hesitant to add any new functionality that may stray too far from the project rubric, so this may be left unoptimized and unimplemented without prior permission from the professor in question.

Second, the author is considering the addition of a set of buttons to the `JTree` panel that would enable users to retrieve information on tree objects highlighted by mouse click. While the search function would be left largely untouched by such functionality, the addition of functionality aimed at allowing users to explore the tree for themselves and retrieve information on selected items would certainly improve all-around usability. Under the present design, the items are properly organized but not very interactive. Whether or not this info retrieval would take the form of more items appended to the search panel or the use of a `JOptionPane`-powered design is yet to be determined. The author is fearful that the appending of items to the right panel may not be readily visible to the user if the user's eyes are focused on the left part of the GUI, hence the consideration of a `JOptionPane`.

Furthermore, the author may add some buttons to permit users to expand all/collapse all tree branches on cue. It is currently a bit irritating to have to hand-open every `SeaPort`

branch for inspection when the addition of a few buttons could automate the operation more legitimately.

Part V: Lessons Learned

In an attempt to make the program output a bit more readable and user-friendly, the author decided to employ a `JTree` implementation that would more reasonably display the data of the associated World object. As a result, the author did a fair bit of personal documentation research into the specific manner in which `JTree` objects can be created, learning much in the process.

Furthermore, the author experimented with several different layouts after the decision to add several more panels to the essential GUI design was reached. After several experiments in moving the options panel to the side or splitting its contents onto two lines rather than one, the author took a step back and considered how a user would want the GUI to be organized for maximum usability. The lesson learned was one of tasteful design when developing something that will be used by someone not necessarily familiar with the workings of the program.