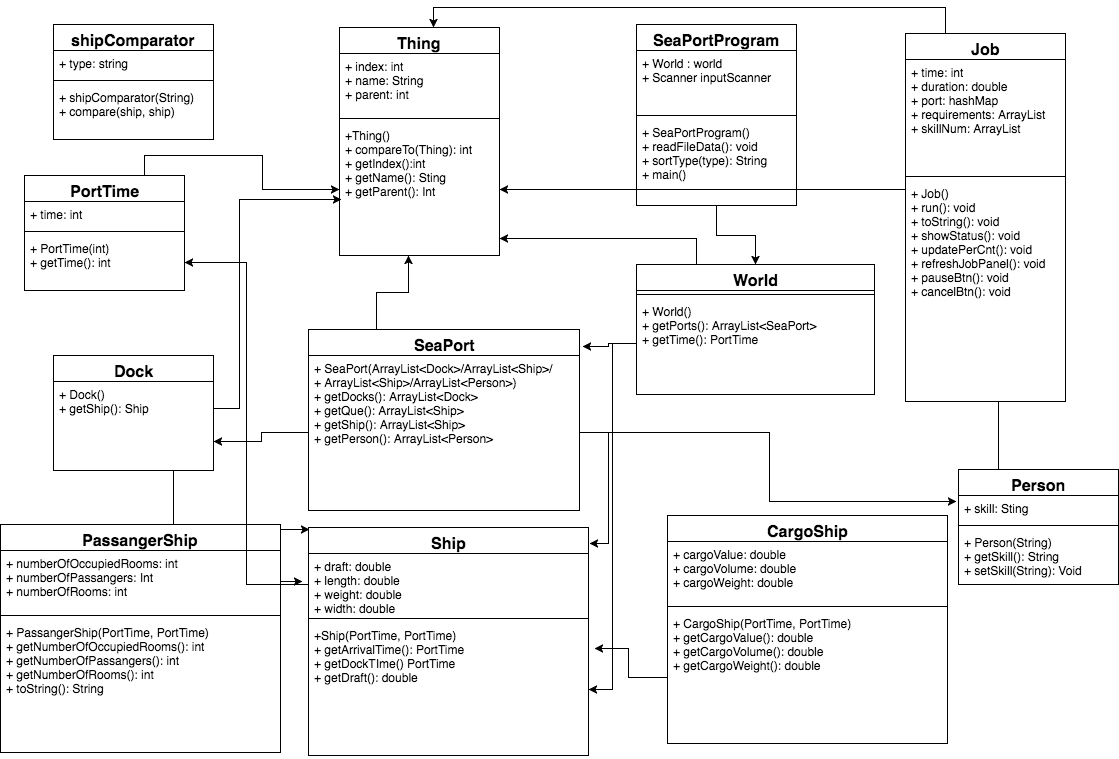
Behrooz Babazadeh

05/02/2017

**Design:**

I wanted to output for the GUI to look better so I cleaned it up a bit. I removed all errors and simplified the data structure. Additionally, implemented threads and resource pools.

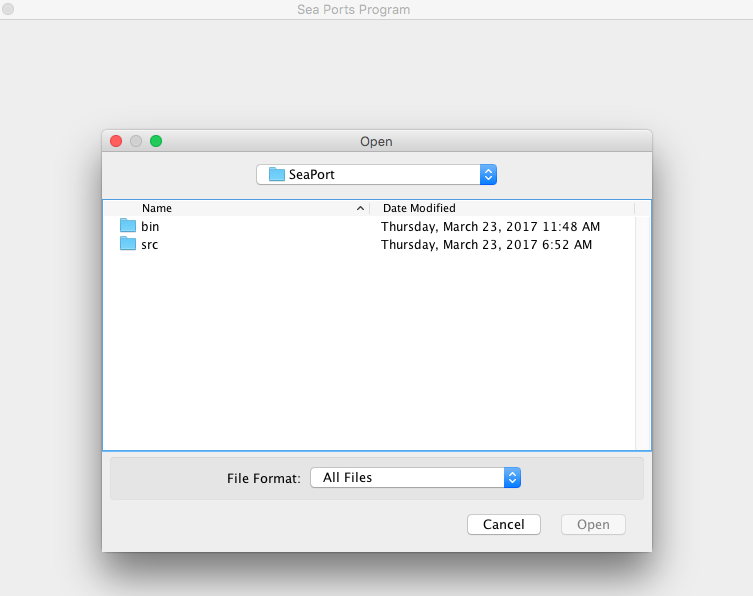
**UML Diagram:**



Note: The Original UML file will be with in the assignment folder. Additionally, I have added the job class and removed some redundant code.

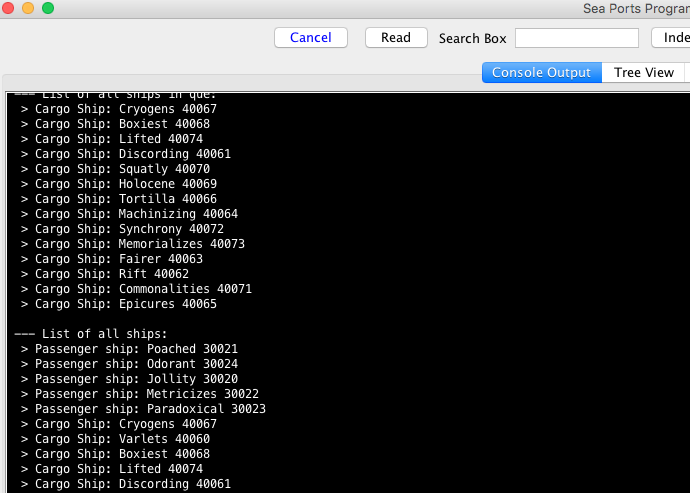
**User guide:**

1. Unzipping the folder and properly implementing the “SeaPortProgram” within your working environment.
2. Simply launch the program, if the program is running correctly (image below).



1. Once the program is running the user will need to pick between the two files that are added with this program (aSPaa.txt and aSpab.txt). The second text file contains the needed job data.

4. Finally once the program reads the proper file and launches. The console output area will print the file data.



**Test Plan:**

The program will utilize the updated job class from the previous update. The class is simpler and better at sorting through the data to execute it’s purpose. Which is to assign jobs to the people and the people data to ships as the tasks complete, update and continue.

**Lessons learned:**

I had a lot to restructure from each class and mainly the job class, which will be tasked in implementing the job/skill data. Even though I know how to implement resource pools, threads and executors due to the structure of between the classes, resulting in errors that needed to be the main focal point.

**Improvements:**

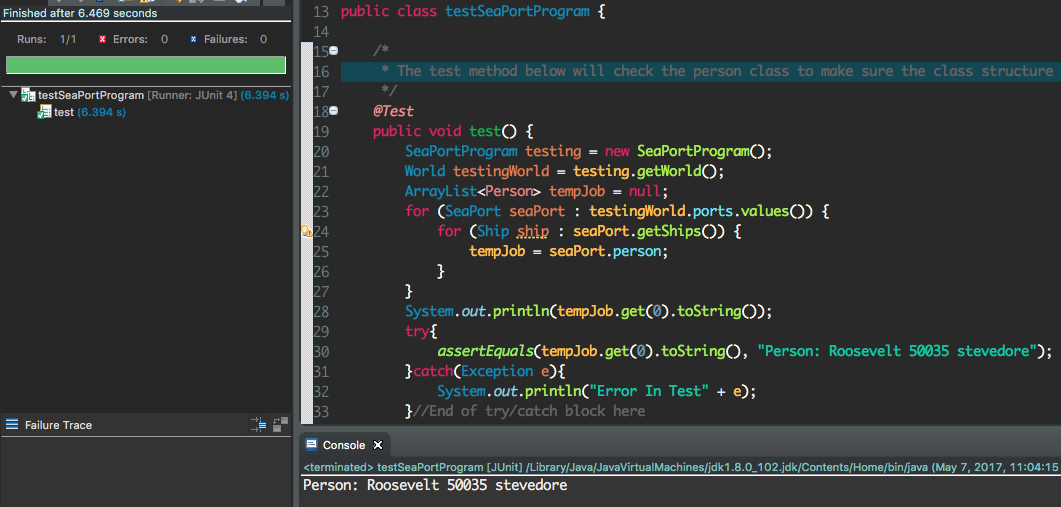
If I had more time I would like to clean up the structure by completely redesigning it but with my current knowlage.

**Test- Cases:**

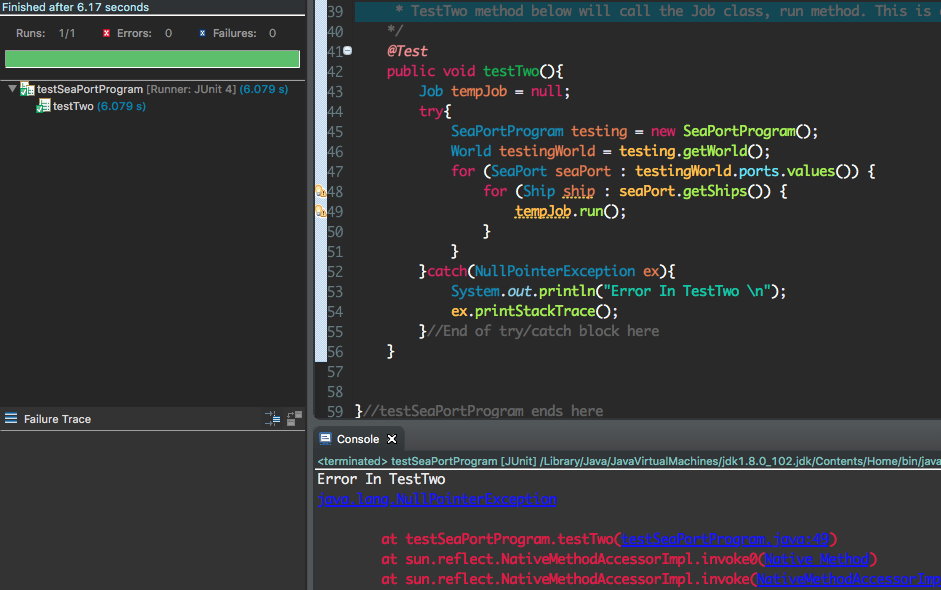
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case**  **#1** | **Input** | **Description** | **Expected Output** | **Actual**  **Output** | **Pass/Fail** |
| JUnit test, Testing the person class after all the new structure changes. | Calls the person class and runs the first element from the variable as a string. | Most of the setters and many getter methods are removed; additionally the structure has been updated. The purpose of this test is to check the person return value for the first element. | The program will return the proper element as a string. | Proper prompt was printed. | Pass |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case**  **#2** | **Input** | **Description** | **Expected Output** | **Actual**  **Output** | **Pass/Fail** |
| JUnit test, Testing if the proper catch error will occur from the test. | The JUnit test calls the job classes run method without proper data. | This test will check if a NullPointerException occurs due to the run method being run without a lot of the data. | I expect the program will toss an error but I am not certain if it will be a NullPointer error. | The program displays the proper output. | Pass |

**Below are the images from the test cases:**



Within the above image, one can see the green pass indicator as well as the proper output statement within the console log.



The image above is a screen shot of TestTwo catching a NullpointerException from the method run. It prints the proper message and it was meant to cause an error from a failure.