Alexander Szelestey

December 13, 2019

Project 4

CMSC 335 7981

**Design**

**UML class diagram**

A close up of a piece of paper

Description automatically generated

**Overview**

The UML diagram that you see above has 13 classes implemented to make seaport Graphical User Interface (GUI) world run. The newest addition is the resourcePools class, and a few modifications to these classes: seaport, seaportProgram, and job. The new class resourcePools creates the panel so we can call on it from the seaportProgram to display how many people with specific skills are currently available. The information is pulled from the SeaPort.ArrayList<Person> to create new object pools that can support the assignment of ship, jobs, and people with skills.

**User's Guide**

**Operations**

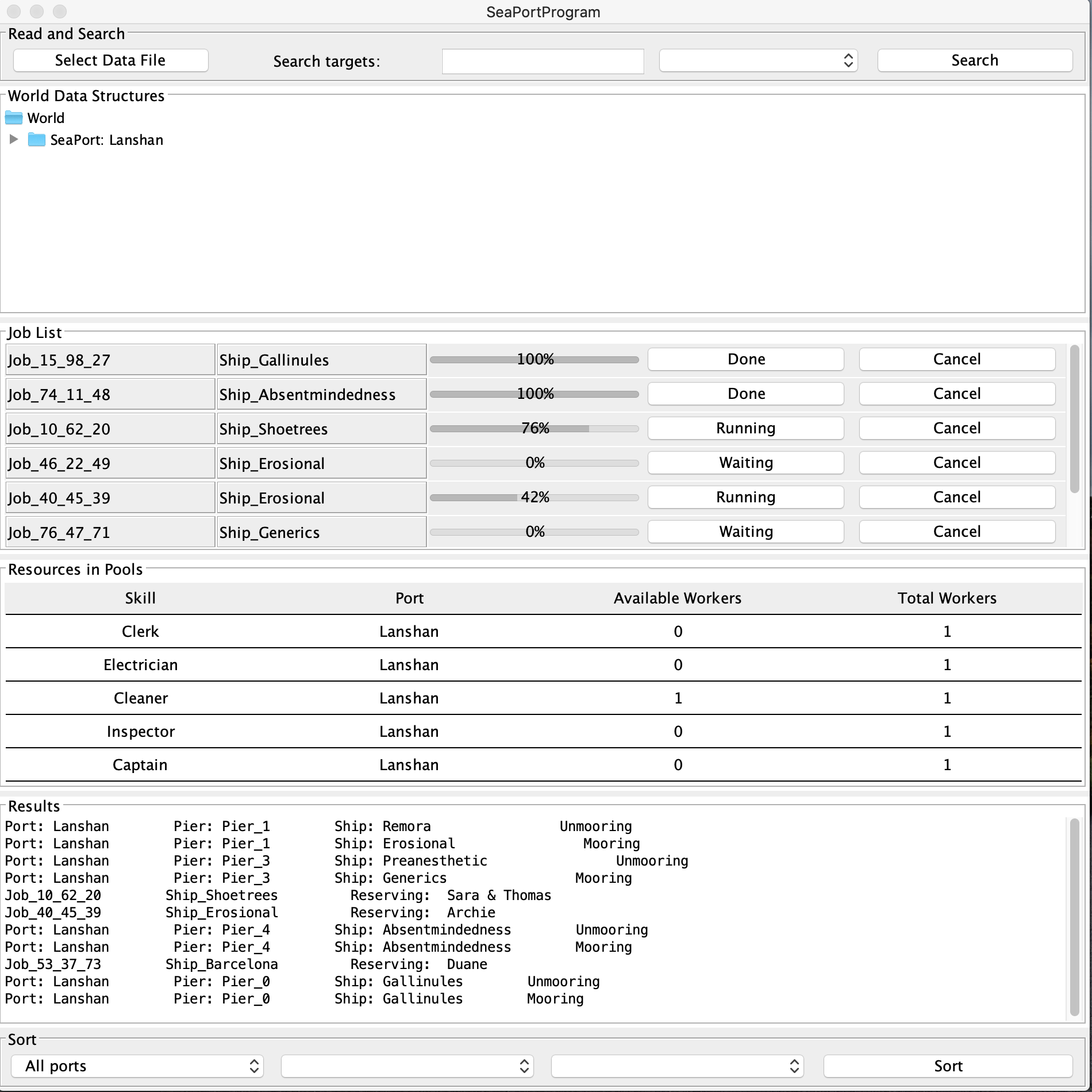
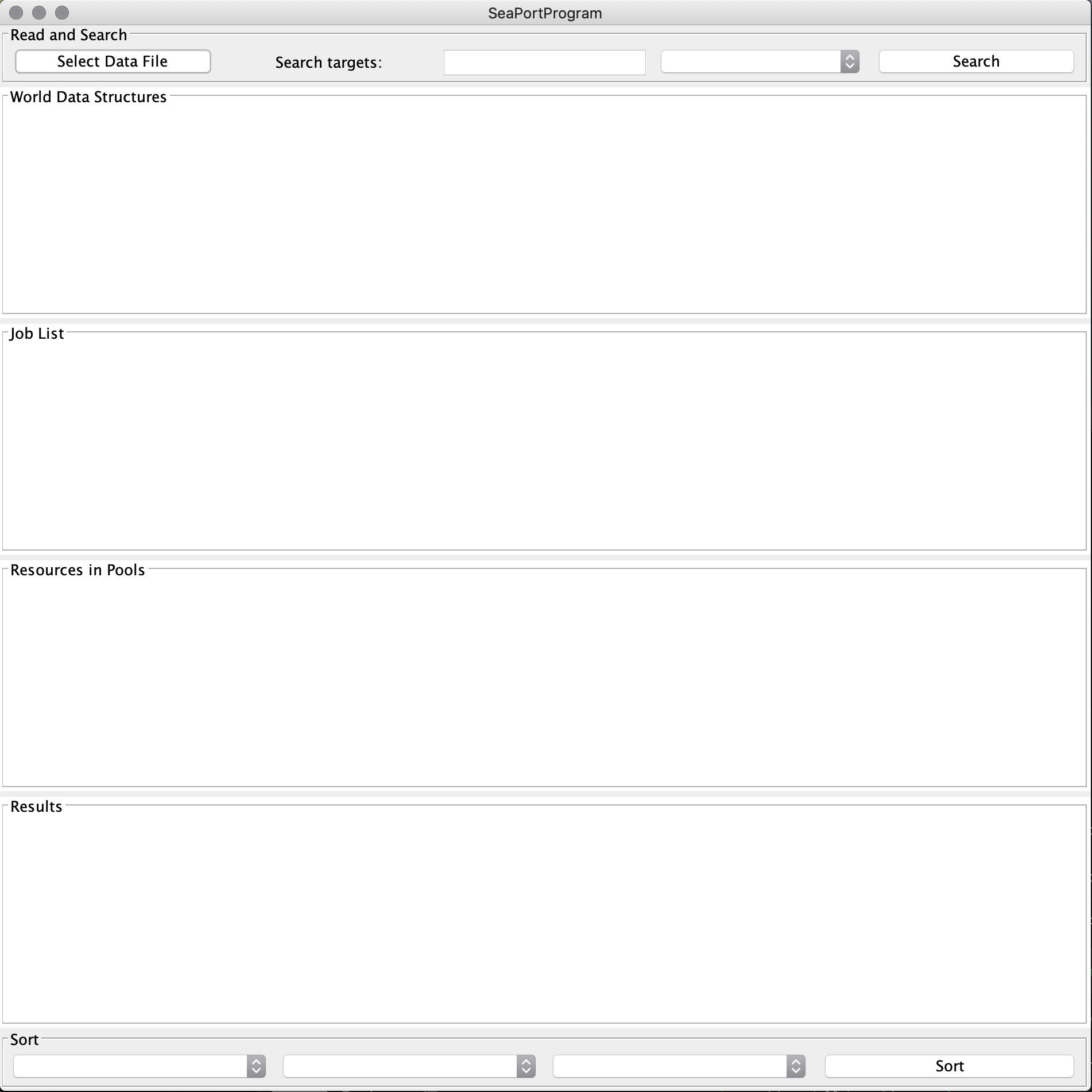
****If you wanted to run this program, having a Java Integrated Development Environment (IDE) would be very beneficial if you didn’t already download one. I recommend Eclipse because it’s easy to use and can add on features. Once the classes are loaded, run configuration and shortly a new interface will pop-up (figure 1.). Click “Select Data File” and choose from preselected text files. There is an “aReadme.txt” and various other files that were given from the class resources. Once you select “aReadme.txt” the user will be prompted with Figure 2. The GUI is broken into 6 different panels: Read and Search, World Data, Job List, Resources in Pools, Results, and Sort. This will help aid the user to find everything they are looking for. In the center is the JProgress bar that allows you to view the status of the jobs being worked on or waiting to be worked on. Below the “Job List” is the “Resources in Pools”; which will constantly update every time a job is completed or starting with the available workers. Lastly, the “Results” which will give an accurate description of the whole seaport. This feed gives like updates on boats coming in and out, jobs starting and finishing, and the search and sort options.

Figure 2.

Figure 1.

**Features**

From Project 3 there a couple of new features that make this GUI visually pleasing. From the last project I changed up the layout, but I reverted it to GridLayout with more panels and labels to help organize the GUI. You could say that adding all these panels is adding in more code than necessary but honestly trying to work with other layouts are just more difficult to implement. The new panel “Resources in Pools” looks like a JTable but it’s made up of 3 panels and 8 labels. Typically, with a JTable, you can add in a column header but for us to get a header I had to duplicate all the classes to return a plain text with a grey background. I enjoy the look of this GUI and how everything is blank before choosing to ‘Select Data File”. Once loaded in the skills, ports, available workers, and total workers are displayed. This gives us an accurate depiction of what skilled worker is at work. Jobs don’t hold any resources if it’s not progressing. The use of synchronization allowed us to avoid race conditions; race conditions are when operations perform simultaneously to produce inaccurate results. Once all jobs are completed the total number of workers match the available workers. This means the methods addPerson/removePerson in ResourcePools is working as it should. To ensure that all ships can port and start/finish their jobs there are synchronized methods in the Jobs class. In the Jobs class, it returns a result which boats are mooring/unmooring at specific ports. Some warnings are populated in the seaport class if the job requires qualified workers or more workers to start a job. Due to some jobs needing more people there is a returning/reserving synchronized method in Job class to aid in first come first serve method. All of this information will be in the “Results” panel.

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

A screenshot of a social media post

Description automatically generated

Figure 5.

Figure 4.

Figure 3.

**Test Plan**

Please use the text files that are included in the project4.zip file to follow along in this test case below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test | Input | Dropdown | Expected Output | Actual Output | Figure |
| 1.  No file loaded click “Search” | “ ” // Blank | Read and Search // blank  Sort // blank | Error - Please Select Data File | Error - Please Select Data File | 3 |
| 2.  No file loaded click “Sort” | “ ” // Blank | Sort // blank | Error – Select Data File to continue | Error – Select Data File to continue | 4 |

“Select Data File” //click

“readme.txt” //select

“Open” // click

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Input | Expected Output | Actual Output | Figure |
| 3. | \*\*Wait for jobs to start and finish  \*\* Some code from “Results” panel | Port: Lanshan  Pier: Pier\_4  Ship: Absentmindedness  Unmooring  Port: Lanshan  Pier: Pier\_4  Ship: Absentmindedness  Mooring  Job\_53\_37\_73  Ship\_Barcelona Reserving: Duane | Port: Lanshan  Pier: Pier\_4  Ship: Absentmindedness  Unmooring  Port: Lanshan  Pier: Pier\_4  Ship: Absentmindedness  Mooring  Job\_53\_37\_73  Ship\_Barcelona Reserving: Duane | 2 |

“Select Data File” //click

“aSPae.txt” //select

“Open” // click

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Input | Expected Output | Actual Output | Figure |
| 4. | \*\*Wait for jobs to start and finish  \*\* Some code from “Results” panel | Port: Kandahar  Pier: Pier\_13  Ship: Abscissions  Unmooring  Ship: Sunnier - needs qualified worker for Job\_64\_50\_19)  Job\_35\_26\_63  Ship\_Magical Reserving: Debbie | Port: Kandahar  Pier: Pier\_13  Ship: Abscissions  Unmooring  Ship: Sunnier - needs qualified worker for Job\_64\_50\_19)  Job\_35\_26\_63  Ship\_Magical Reserving: Debbie | 5 |

**Lessons Learned**

A screenshot of a cell phone

Description automatically generatedIf I had more time, In the “Resources in Pool” panel I would want to figure out how to keep the column header stationary and have the JScroll on a separate panel so while you scroll you can still read the header. I tried implementing JSplitPanel but was unsuccessful and I probably just had the code positioned in the wrong area. Maybe it wasn’t the best way to implement what I was trying to do but that’s all I found on stack overflow and setting preferred size didn’t work either. I undid it all and left it the way it is represented in the pictures. (If you scroll, the headers disappear like in Figure 6.)

Figure 6.

The best way to implement a counting method for Available workers was to use a HashMap that I found in the comments section of stack overflow (Gregory & Russter, 1958). I had to modify it from what they had originally implemented it but instead of brackets I used parenthesis and the errors went away. The code is in the Class SeaPort lines 97-99

Notice how in Figure 6 underlines are separating each row. Originally everything hard borders but I was able to figure out how to only make certain borders appear, curtesy again by stack overflow (Becky, 1960). I believe it makes the final product very clean and easy to read. The code is in the Class SeaPortProgram lines 163-166.

I got a better understanding of object pools, concurrent collections, synchronization, GUI handling, and multiple little things that are starting to click the more I work with Java and programming in general. There were a lot of situations from the beginning that were confusing and having the course resources and online forums like stack overflow and GitHub help make the connection. It’s easy to read and understand the purpose of synchronization and concurrency on oracle, but it is very difficult to implement them into your code. Luckily for me and I m assuming other students have these forums to help us better understand what we are trying to implement and run.

References

Becky. (1960, March 01). Is it possible to have a java swing border only on the top side? Retrieved December 13, 2019, from <https://stackoverflow.com/questions/2174319/is-it-possible-to-have-a-java-swing-border-only-on-the-top-side>

Eissen A. (2018, March 10). Seaport\_project\_3. Retrieved December 1, 2019, from <https://github.com/andreweissen/SeaPort_Project_2>

Gregory, & Russter. (1958, November 01). Most efficient way to increment a Map value in Java. Retrieved from <https://stackoverflow.com/questions/81346/most-efficient-way-to-increment-a-map-value-in-java>

Karen. (1959, December 01). Setting the size of panels. Retrieved December 14, 2019, from <https://stackoverflow.com/questions/1594423/setting-the-size-of-panels>

Paul, J. (2017, February 6). What is blocking methods in Java and how do deal with it? Retrieved December 14, 2019, from <https://javarevisited.blogspot.com/2012/02/what-is-blocking-methods-in-java-and.html#:~:targetText=As%20I%20said%20Blocking%20methods,until%20reading%20of%20file%20finished>.

ResourcePool. (2015, August 26). Retrieved December 14, 2019, from <https://docs.oracle.com/middleware/1221/adf/api-reference-model/oracle/jbo/pool/ResourcePool.html>

Sugianto, R. (1960, April 01). What does the colon (:) operator do? Retrieved December 13, 2019, from <https://stackoverflow.com/questions/2399590/what-does-the-colon-operator-do>