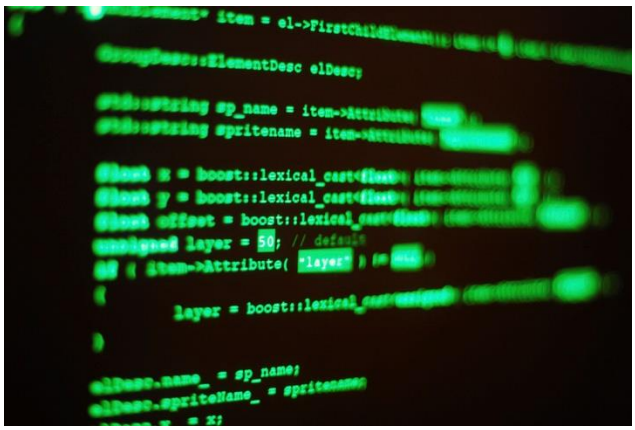
Project 4: The SeaPort Project Series

By: Rebecca Davis

CMSC 335

Due: March 4, 2018



Project 4: The SeaPort Project series

Rebecca Davis
CMSC 335
Due Date: March 4, 2018

Program Description

This program is designed to read data from a text file selected by the user and create an internal data structure representing the elements in the file and their relationships. The program, which features a GUI, allows the user to *search* or *sort* the elements of the structure using various criteria. The user must select a text file that has previously been created in the required format for this program. Upon starting the program, a `JFileChooser` pops up immediately allowing the user to make this selection. This text file is then used to create the internal data structure to store the items from the file. A `JTree` representing the data structure is created using the objects generated and it is displayed on the left side of a `JSplitPane` under a tab labeled 'Data Structure'. The GUI features `JRadioButtons` to select whether to perform a search or sort operation, a search bar, and two drop-down menus pertaining to the criteria options the user has. The results of the search operation are displayed on the right side of the split pane in a text area, under the 'Data Structures' tab. The results of the sort operation are displayed in this same area, as well as in the `JTree`, which is redrawn once the sort operation is performed.

If the user attempts to enter something into the GUI that is not found in the text file or if the input field is left blank, an error message informing them of this will be displayed. The user can enter strings of various cases and the program will still recognize what to search for even if the input is not in the correct case—for example, if the user would like to search for the person "Leroy," entering "LERoy" will not cause an error. The results for Leroy will still be displayed. Thus, the program is designed to ignore the case of user input.

Project 2 of the SeaPort Project series expanded on the foundation set in place by the first project. This project introduced a new advanced data structure – hash maps. The `HashMap` class was utilized to implement linking the classes created in Project 1. The hash maps are local to the method used to read the text file, which is the `fillWorld()` method located in the `World` class. The index and object information are placed into a `HashMap` for that object. The parent index is then grabbed from the corresponding parent `HashMap`. Then, the index is added to the `ArrayList` for the object. As a result of this change in how classes are linked, the index and parent variables, once located in Thing class, were removed because they were no longer needed by the program. In addition, many methods in this program were changed or removed. For example, the methods to link object data (e.g., `assignShip`, `assignDock`, `assignPerson`), which utilized other methods designed to get information by index number (e.g., `getShipByIndex`, `getPersonByIndex`, `getDockByIndex`) were all removed.

Another noteworthy change to Project 2 was that comparators were implemented in order to support the operation of sorting. Comparator classes were added to the file of the class corresponding to each object and they each contain a `compare()` method to sort the aspect of the object being compared. Comparators are contained in the following classes: `Ship`, `Person`, and `Job`. Several sort methods for the objects are also created and they exist in the class in which their array lists are located. Thus, these sort methods are located within the following classes: `World`, `SeaPort`, and `Ship`. The `Collections.sort()` method is used in these methods and new comparator objects are also created here. The `World` class contains higher-level sort methods that call these methods just described.

Project 3 of the SeaPort Project series further expanded upon the first two projects. This project made uses of the Swing class `JTree` to display the data structure of the text file. The

`JTree` is drawn immediately after the text file is read. Additionally, multithreading was introduced in this project. A thread was implemented for each job representing a task that its ship requires. The `synchronized` directive offered in Java was utilized in order to avoid potential race conditions that could occur and namely, to ensure that a dock is performing the jobs for only one ship at a time. Therefore, none of the jobs of a ship in the queue for a dock are in progress. When all of the jobs for a ship are complete, that ship leaves the dock in order for a ship in the dock's queue to be docked. Once docked, the new ship's jobs can begin to progress. The Swing class `JProgressBar` was used to display the progress of the jobs. There are also `JButtons` on each job panel to allow a job to be suspended or cancelled. Finally, a `JTabbedPane` was added to the GUI in order to display the 'Data Structure' and 'Job Progress' separately. Please see the *Design* section in this document for a list of all of the changes, additions, and removals that were made to Project 3.

For the final round of this project series, Project 4 adds complexity to the threading process by considering the *requirements* for the jobs. In particular, jobs are instructed to wait until people with the skills required by the job were available at the port. Functionality to read job requirements from the text file and adding them to each Job instance was already implemented in a previous project. However, for Project 4, the `ArrayList` for persons is treated as a *resource pool*, along with supporting assignment to ships and jobs. The multithreading capability in Project 3 is expanded with the use of these resource pools and this demonstrates the capability of the program to blocks certain threads until the required resources become available. A job is instructed to start if its ship is at a dock and all necessary persons with the required skills are available. Otherwise, the job releases those resources if it cannot progress. `Synchronization` directives were utilized to avoid race conditions. Each job thread holds

onto required `synchronization` locks for a short period of time. When a job is over, the resources used by that job (i.e. persons) are released back to the port. When all of the jobs of a ship are complete, the ship leaves the dock and one of the ships in que is assigned to a free dock so that it can begin to process its jobs. If a job can never progress because the port does not have the necessary persons (with the skills reflecting the requirements for the job) at the port, the job is cancelled. GUI appearance was enhanced greatly in Project 4. Under the 'Job Progress' tab, there are two `JTables` separated by a `JSplitPane`. The 'Jobs' table displays each job and its respective port, ship, requirements, job progress, and resources (people) acquired. The 'People' table displays each person according to the port to which they belong, the skill they have, and the ship at which they are currently working (when applicable).

The program contains several classes. The main class, `SeaPortProgram4`, defines the GUI. It contains a constructor to build the GUI, event handlers that define instructions for the *Search/Sort,* and *Start* buttons, and the main method which runs the GUI. The second class, Thing, implements Comparable<Thing>. It is the parent class to several other classes—`World`, `SeaPort`, `Dock`, `Ship`, `Person`, and `Job`. The `Ship` class also has two child classes—`PassengerShip` and `CargoShip`—which represent two types of ships. Each class contains a scanner constructor that allows the class to use super constructors while adding any unique elements that are particular to only that class. Each class also uses the `toString()` method in the `Thing` class to display the object name and a contains its own `infoString()` method used to display the remainder of the information for that object. The `World` class is unique in that it comprises several methods not found in the other classes. For example, it includes a method to read the text file line by line, create the `HashMaps` and add the indices to the `ArrayLists`, several methods to find information in the file by certain criteria (e.g.,

`findJobByRequirement`, `findShipByName`, `findPersonBySkill`), as well as the higher-level sorting methods outlined earlier. As was true for Project 1 and Project 2, the `World` class contains a bulk of the methods applied in the SeaPort program. The `Job`, `World`, and `SeaPort` classes were enhanced a great deal to incorporate methods to perform the multithreading process.

Design:

Classes, Fields, Constructors, and Methods:

The following list outlines the class structure for SeaPortProgram4, including the fields, constructors, and methods. A description of each is included.

**The items in green represent additions made specifically to SeaPortProgram4 for Project 4 of the SeaPort Project series. Removed fields and methods are listed at the end of this section.*

Class: SeaPortProgram4 (extends JFrame)

Fields:

- GUI interface variables – *essential to the SeaPortProgram4 GUI*

(**some fields related to the GUI were added or changed to expand functionality of GUI*)

- World world – *World object; contains information pertaining to World*

Constructors:

- SeaPortProgram4() – *class constructor for the GUI*

Methods:

- setFrame() – *method to set the frame and other features of the GUI*
- typeComboBoxActionPerformed() – *event handler for 'Type' combo box*

- updateOptionList() - *method to update options to combo boxes based on radio button (Search vs. Sort) and Type selected*
- sortRadioButtonActionPerformed() - *event handler for 'Sort' radio button*
- searchRadioButtonActionPerformed() - *event handler for 'Search' radio button*
- searchButtonActionPerformed() – *event handler for 'Search' button*
- progressButtonActionPerformed() – *event handler for 'Progress button*
- drawTree() – *method to create JTree to display file structure; contains a TreeSelectionListener() to display information based on what user has selected*
- createTreeNodes(String title) – *method to create the nodes that are used to populate the JTree*
- addNewTreeBranch(ArrayList<T> thingList, String name) – *method to add a new branch to the JTree*

Class: Thing (implements Comparable<Thing>)

Fields:

- String name – *variable to hold name*
- Thing parent – *Thing object to hold parent information*

Constructors:

- Thing(Scanner scanner, String name, Thing parent) – *scanner constructor; parameter added (Thing parent)*
- Thing() – *empty, no-argument constructor*

Methods:

- compareTo(Thing o) – *method for comparison*
- toString() – *method to return a string containing name only (parent removed)*
- infoString() – *method to return a string containing information; replaces previous toString() method*
- getName() – *method to return name*
- getParent() – *method to return parent*

Class: World (extends Thing)

Fields:

- ArrayList<SeaPort> ports – *ArrayList to hold sea port information*
- JTable peopleTable – *JTable to display job threads information in the 'People' table*
- JTable jobsTable – *JTable to display job threads information in the 'Jobs' table*
- Object[ ] [ ] peopleTableData – *2D object array to hold job data for 'People' table*
- Object[ ] [ ] jobsTableData– *2D object array to hold persons data for 'Jobs' table*
- TableModelClass peopleModel – *object of class used for table*
- TableModelClass jobsModel – *object of class used for table*

Constructors:

- World(Scanner scanner, String name, Thing parent) – *scanner constructor; parameter added (Thing parent)*
- World() – *empty, no-argument constructor*

Methods:

- fillWorld(Scanner scanner) – *method that takes the lines from scanner and creates **HashMaps** (*HashMaps are local to this method) used to the link classes together*
- startJobs(JPanel progressPanel) - *method to start threading process of jobs in file*
- infoString() – *method to return a string containing information; replaces previous toString() method*
- findPersonByName(String name) – *method to find Person information by name*
- findPersonBySkill(String skill) – *method to find Person information by skill*

- findJobByName(String name) – *method to find Job information by name*
- findJobByRequirement(String requirement) – *method to find Job information by requirement*
- findShipByName(String name) – *method to find Ship information by name*
- findPortByName(String name) – *method to find Port information by name*
- findPortByShip(String ship) – *method to find Port information by ship name*
- findPortByPerson(String person) – *method to find Port information by person name*
- findDockByName(String name) – *method to find Dock information by name*
- sortPortsByName() – *method to sort Ports by name*
- sortDocksByNamePorts() – *calls sortDocksByName() method in SeaPort class*
- sortPeopleByNamePorts() – *calls sortPeopleByName() method in SeaPort class*
- sortPeopleBySkillPorts() – *calls sortPeopleBySkill() method in SeaPort class*
- sortJobsByNameShips() – *calls sortJobsNameAllShips() method in SeaPort class*
- sortJobsByDurationShips() – *calls sortJobsDurationAllShips() method in SeaPort class*
- sortQueByNamePorts() – *calls sortQueByName() method in SeaPort class*
- sortQueByDraftPorts() – *calls sortQueByDraft() method in SeaPort class*
- sortQueByLengthPorts() – *calls sortQueByLength() method in SeaPort class*
- sortQueByWeightPorts() – *calls sortQueByWeight() method in SeaPort class*
- sortQueByWidthPorts() – *calls sortQueByWidth() method in SeaPort class*
- sortAllShipsByNamePorts() – *calls sortAllShipsByName() method in SeaPort class*
- sortAllShipsByDraftPorts() – *calls sortAllShipsByDraft() method in SeaPort class*
- sortAllShipsByLengthPorts() – *calls sortAllShipsByLength() method in SeaPort class*
- sortAllShipsByWeightPorts() – *calls sortAllShipsByWeight() method in SeaPort class*
- sortAllShipsByWidthPorts() – *calls sortAllShipsByWidth() method in SeaPort class*

- getPorts() – *method to return ports*
- updateTable(int row, int col) – *method to update table; setValueAt(), fireTableDataChanged(),and repaint() methods called here*
- updatePersonTable(int row) – *method to update second table; setValueAt(), fireTableDataChanged(),and repaint() methods called here*

Class: TableModelClass (extends DefaultTableModel)

Fields:

- *serialVersionUID – necessary to avoid/remove compiler warning*

Constructors:

- *TableModelClass(Object[][] objectAray, String[] stringArray) – Constructor*
- *TableModelClass() – Constructor*

Methods:

- *getColumnClass(int c) – Method to get column class*

Class: CustomContainerRenderer (implements TableCellRenderer)

Fields:

- *serialVersionUID – necessary to avoid/remove compiler warning*

Methods:

- *getTableCellRendererComponent() – Method that returns component of table cell*

Class: JTableButtonMouseListener (extends MouseAdapter)

Fields:

- *JTable table – JTable variable*

Constructors:

- *JTableButtonMouseListener(JTable table) – Constructor*

Methods:

- *mouseClicked(MouseEvent e) – Method to handle mouse clicks; forwards mouse clicks to JButtons*

Class: SeaPort (extends Thing)

Fields:

- ArrayList<Dock> docks – *ArrayList to hold docks*
- ArrayList<Ship> que (list of ships waiting to dock) – *ArrayList to hold ships waiting to dock*
- ArrayList<Ship> ships (list of all the ships at this port) – *ArrayList to hold all ships at a port*
- ArrayList<Person> persons (people with skills at this port) – *ArrayList to hold people with skills at a port*

Constructors:

- SeaPort(Scanner scanner, String name, Thing parent) – *scanner constructor; parameter added (Thing parent)*

Methods:

- infoString() – *method to return a string containing information; replaces previous toString() method*
- sortPeopleByName() – *method to sort People by name*
- sortPeopleBySkill() – *method to sort People by skill*
- sortJobsNameAllShips() – *method to sort Jobs by name*
- sortJobsDurationAllShips() – *method to sort Jobs by duration*
- sortQueByName() – *method to sort Ships in Que by name*
- sortQueByDraft() – *method to sort Ships in Que by draft*
- sortQueByLength() – *method to sort Ships in Que by length*

- sortQueByWeight() – *method to sort Ships in Que by weight*
- sortQueByWidth() – *method to sort Ships in Que by width*
- sortAllShipsByName() – *method to sort all Ships by name*
- sortAllShipsByDraft() – *method to sort all Ships by draft*
- sortAllShipsByLength() – *method to sort all Ships by length*
- sortAllShipsByWeight() – *method to sort all Ships by weight*
- sortAllShipsByWidth() – *method to sort all Ships by width*
- sortDocksByName() – *method to sort Docks by name*
- getDocks() – *method to return docks*
- getQue() – *method to return ships in que*
- getShips() – *method to return all ships*
- getPersons() – *method to return persons*
- getNextShipInQue() – *method to return the next ship in que*
- findAvailablePersons(String skill) – *method to find available person for job*
- getPeopleForRequirements(ArrayList<String> requirements) – *method to get people with skills that match job's requirements*
- areRequirementsCovered(ArrayList<String> requirements) – *method to check whether all requirements are covered for a job (i.e., that there are enough resources to perform the job)*

Class: PersonSkillComparator (implements Comparable<Person>); moved from Person class to address compiler warning from Project 3

Class: ShiftDraftComparator (implements Comparable<Ship>); moved from Ship class to address compiler warning from Project 3

Class: ShiftLengthComparator (implements Comparable<Ship>); moved from Ship class to address compiler warning from Project 3

Class: ShiftWeightComparator (implements Comparable<Ship>); moved from Ship class to address compiler warning from Project 3

Class: ShiftWidthComparator (implements Comparable<Ship>); moved from Ship class to address compiler warning from Project 3

Class: Dock (extends Thing)

Fields:

- Ship ship – *Ship object; contains information pertaining to a ship (not an ArrayList because there is only one ship at a dock)*

Constructors:

- Dock(Scanner scanner, String name, Thing parent) – *scanner constructor; parameter added (Thing parent)*

Methods:

- infoString() – *method to return a string containing information; replaces previous toString() method*
- getShip() – *method to return ship*
- processNextShip() – *method to change the ship at the dock to one waiting in the queue*

Class: Ship (extends Thing)

Fields:

- double draft – *variable to hold draft value of ship*
- double length – *variable to hold length value of ship*
- double weight – *variable to hold weight value of ship*
- double width – *variable to hold width value of ship*
- ArrayList<Job> jobs – *ArrayList to hold jobs*

Constructors:

- Ship(Scanner scanner, String name, Thing parent) – *scanner constructor; parameter added (Thing parent)*

Methods:

- infoString() – *method to return a string containing information; replaces previous toString() method*
- sortJobsByName() – *method to sort Jobs by name*
- sortJobsByDuration() – *method to sort Jobs by duration*
- changeParent() – *method to change parent of the ships moving from queue to dock*
- getPort() – *method to return port*
- isShipFinished() – *method to return whether a ship is finished processing all jobs*
- isShipReady() – *method to return whether a ship is ready to process jobs*

Class: JobDurationComparator (implements Comparable<Job>); moved from Job class to address compiler warning from Project 3

Class: PassengerShip (extends Ship)

Fields:

- int numberOfOccupiedRooms – *variable to hold # of occupied rooms*
- int numberOfPassengers – *variable to hold # of passengers*
- int numberOfRooms – *variable to hold # of rooms*

Constructors:

- PassengerShip(Scanner scanner, String name, Thing parent) – *scanner constructor; parameter added (Thing parent)*

Methods:

- infoString() – *method to return a string containing information; replaces previous toString() method*

Class: CargoShip (extends Ship)

Fields:

- double cargoValue – *variable to hold cargo value*
- double cargoVolume – *variable to hold cargo volume*
- double cargoWeight – *variable to hold cargo weight*

Constructors:

- CargoShip(Scanner scanner, String name, Thing parent) – *scanner constructor; parameter added (Thing parent)*

Methods:

- infoString() – *method to return a string containing information; replaces previous toString() method*

Class: Person (extends Thing)

Fields:

- String skill – *variable to hold skill*
- boolean isWorking – *variable to hold whether person is working on a job*
- Ship ship – *Ship object; represents ship that person is working on*
- int rowTable – *variable to hold row of table where data is held*

Constructors:

- Person(Scanner scanner, String name, Thing parent) – *scanner constructor; parameter added (Thing parent)*

Methods:

- infoString() – *method to return a string containing information; replaces previous toString() method*
- assignShip(Ship ship) – *method to assign ship and set person to working*

- unassignShip() – *method to unassign ship and set person to not working*
- getShipName() – *method to return ship name*
- getIsWorking() – *method to return whether person is working on a job*
- getSkill() – *method to return skill*
- setRow(int rowTable) – *method to set rowTable value*

Class: Job (extends Thing implements Runnable)

Fields:

- double duration – *variable to hold duration of job*
- ArrayList<String> requirements – *variable to hold requirements of the job (overlaps with some of the skills of the people)*
- boolean suspendFlag – *variable used to suspend a job*
- boolean cancelFlag – *variable used to cancel a job*
- JButton statusButton – *variable for status button*
- JButton cancelButton – *variable for cancel button*
- JProgressBar progressBar – *variable for progress bar*
- Object[] tableRow – *object array to hold data for row of table*
- int rowTable – *variable to hold row of table where data is held*
- enum Status {RUNNING, SUSPENDED, WAITING, DONE} – *Enum type to denote status of thread*
- Status status – *Status variable*

Constructors:

- Job(Scanner scanner, String name, Thing parent) – *scanner constructor; parameter added (Thing parent)*

Methods:

- setTableObject(Object[] object, int rowTable) – *method to set up table object to display job threading data*
- statusButtonActionPerformed() – *event handler for status button*
- cancelButtonActionPerformed() – *event handler for cancel button*
- startJob() – *method to start a job*
- run() – *method to handle threads; makes use of synchronized directive*
- toggleSuspendFlag() – *method to set suspendFlag to true*
- toggleCancelFlag() – *method to set cancelFlag to true*
- showStatus() – *method to show job status*
- isFinished() – *method to return whether a job is finished*
- isBusy() – *method to return whether a job is busy running*
- infoString() – *method to return a string containing information; replaces previous toString() method*

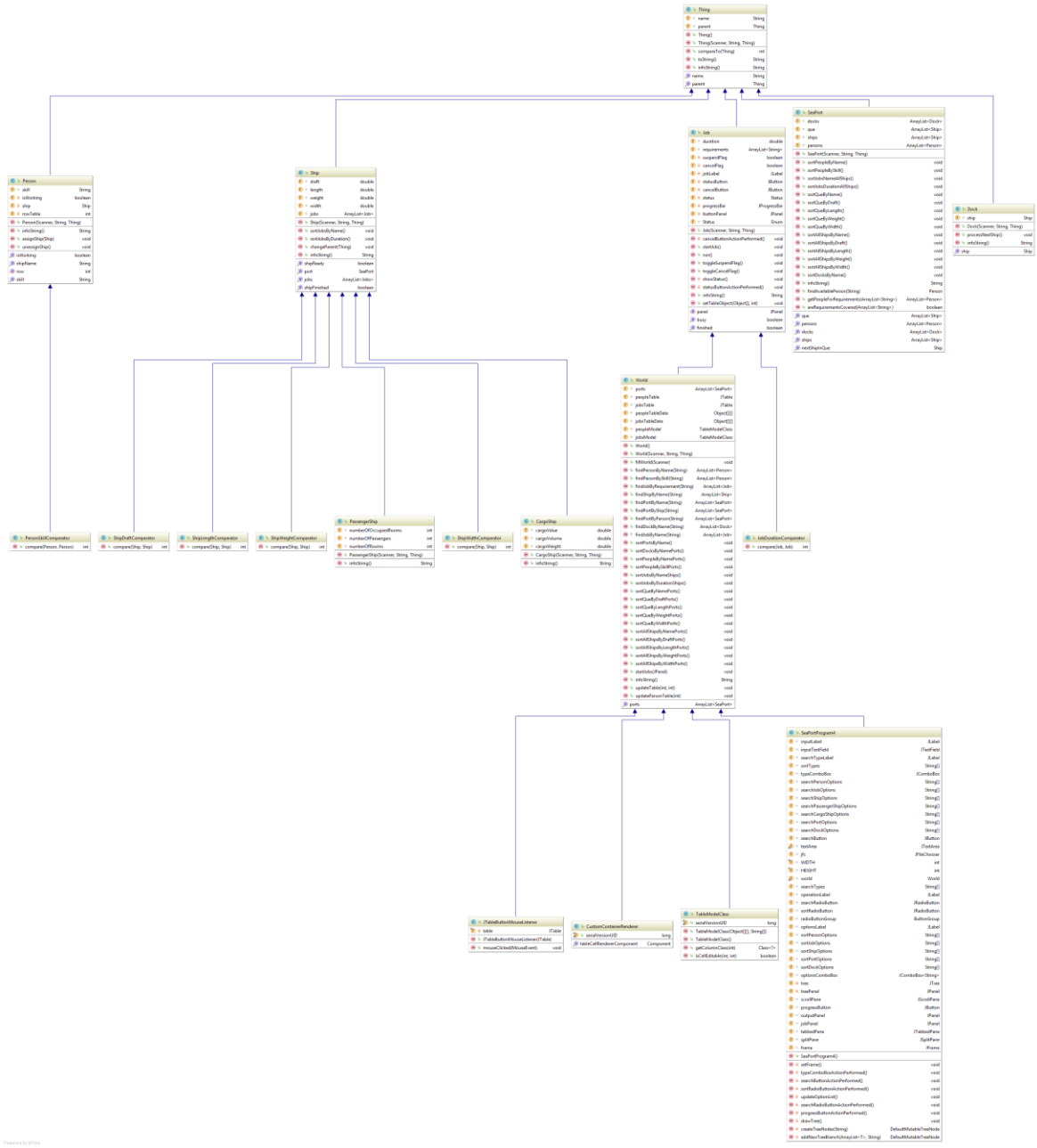
***Removed items from SeaPortProgram:**

Class – Job:

- JLabel jobLabel – *variable for job name*
- JPanel buttonPanel – *variable for button panel; contains status and cancel buttons*
- setPanel() – *method to set up panel displaying progress of jobs*

UML Diagram:

The following is an updated UML diagram, which represents the structure of SeaPortProgram4. This diagram reflects all of the changes to the fields and methods of the classes, as well as any additional classes that were added. It is also attached as a separate file with the project submission.



User's Guide:

The SeaPort application will allow you to choose a text file of a specified format (described below), which will display all contents of the data structure of the file in a tree format. The GUI will also allow you to search various criteria within the file and/or sort the contents of the file by chosen criteria. Before delving into the details on how to use this application, it is important to note that it requires the creation of a text file with information in the following format:

Sample text file *(*note: file has been shortened below to save space)*

```
// File: aSPad.txt
// Data file for SeaPort projects
// Date: Wed Jan 10 20:08:46 EST 2018
// parameters: 8 15 20 20 5 30
//   ports, docks, pships, cships, jobs, persons

// port   name index parent(null)
//    port   <string> <int> <int>
port Ashgabat 10001 0
port Fremont 10000 0
port Vavau 10005 0

// dock   name index parent(port)
//    dock   <string> <int> <int>
  dock Pier_20 20020 10001 30031
  dock Pier_42 20042 10002 30056
  dock Pier_46 20046 10003 30063
  dock Pier_9 20009 10001 30020
  dock Pier_49 20049 10004 30081
  dock Pier_12 20012 10001 30023

// 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         Gatecrashers 30139 10006 147.18 299.62 33.47
23.22 2290 738 738
    pship            Chopping 30045 20031 64.00 105.99 119.92
16.57 660 174 174
```

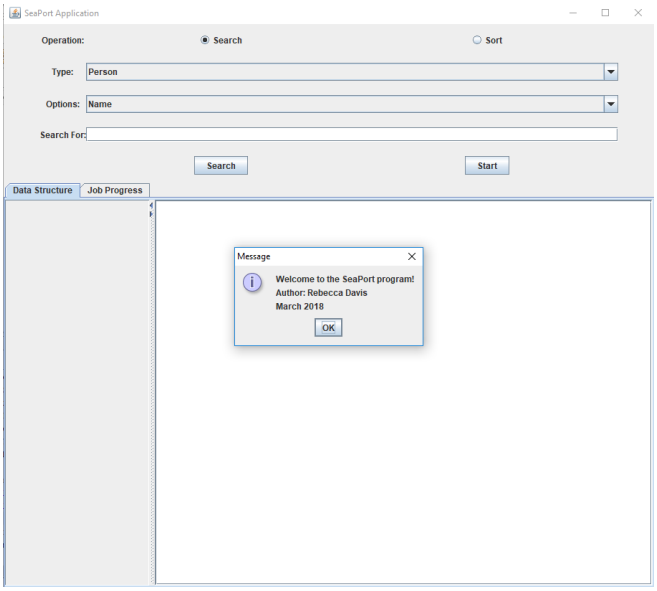
```
    pship            Pillories 30048 20034 220.59 138.52 85.78
38.29 336 149 149
    pship        Valedictories 30057 10002 86.41 344.89 122.64
32.85 1004 1071 502
    pship            Lectures 30036 20025 90.66 154.96 55.84
39.88 3253 1044 1044
    pship       Decontaminated 30083 20051 219.60 230.01 97.85
42.46 3512 1090 1090

// 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               Bassly 40116 10005 94.36 483.30 93.68
37.25 115.46 158.50 618.83
    cship                Valve 40097 10004 151.88 305.45 49.65
44.57 133.54 148.64 885.49
    cship            Generates 40113 10005 199.82 398.76 91.61
36.24 198.97 101.72 359.34
    cship            Alluvial 40064 10003 147.94 236.84 115.46
33.62 99.18 156.63 880.11
    cship              Cancans 40087 10004 172.10 154.48 125.46
23.88 190.17 179.50 782.01

// person name index parent skill
//    person <string> <int> <int> <string>
    person                 Elsa 50043 10004 captain
    person                Leroy 50058 10006 craneOperator
    person              Wallace 50053 10005 electrician
    person             Angelica 50030 10002 craneOperator
    person               Nellie 50008 10001 clerk
    person               Marsha 50050 10005 cleaner
    person                 Kirk 50055 10005 captain
    person                 Rose 50015 10001 craneOperator
    person                  Guy 50048 10005 electrician

// job    name index parent duration [skill]+ (one or more,
matches skill in person, may repeat)
//    job    <string> <int> <int> <double> [<string>]+
    job         Job_39_41_99 60015 30017 26.81
    job         Job_41_52_51 60372 30142 63.21
    job         Job_22_81_90 60130 40044 41.74 mechanic cleaner
electrician
    job         Job_19_17_16 60368 30140 63.57 cleaner
craneOperator
    job         Job_40_61_38 60199 40077 41.42 painter mechanic
mechanic
```

```
    job         Job_34_81_24 60098 40039 64.05 painter mechanic
driver
    job         Job_83_72_48 60261 30101 76.27 electrician
craneOperator crew
    job         Job_71_60_78 60228 30093 97.70 crew painter
    job         Job_79_42_20 60286 30119 62.23 mate mechanic
    job         Job_10_77_37 60252 40092 115.06 carpenter
```

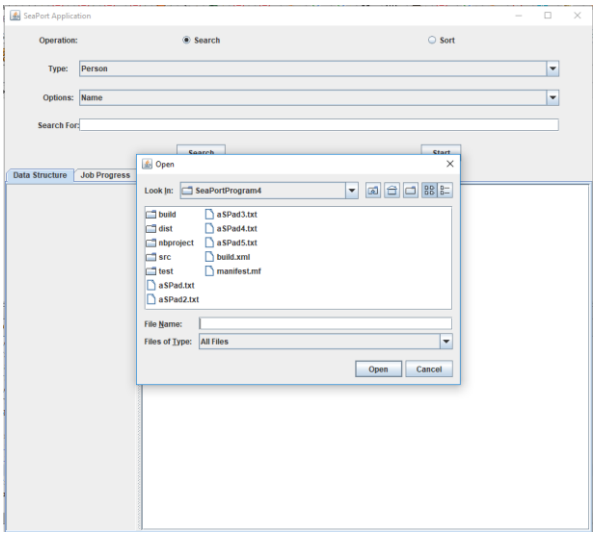
**The test files for this program can be created using Professor Duchon's CreateSeaPortDataFile program.*

To start the program, first obtain the source code. Once obtained, compile and run the code in the command prompt or in your preferred IDE.

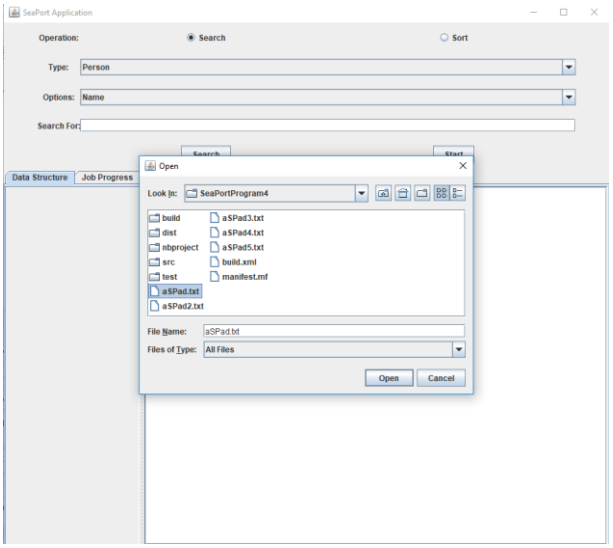
Once the program beings, you will see the following welcome display:



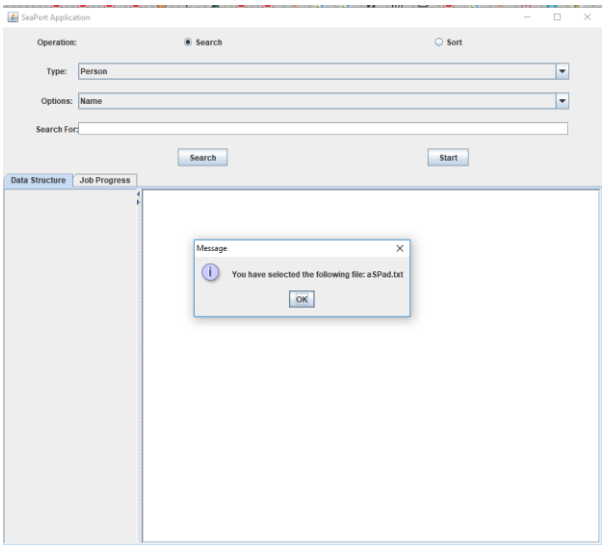
Click ‘OK’ and a dialog box will open to allow you to select a text file from the directory.



Select a text file that contains the appropriate format and click ‘Open.’

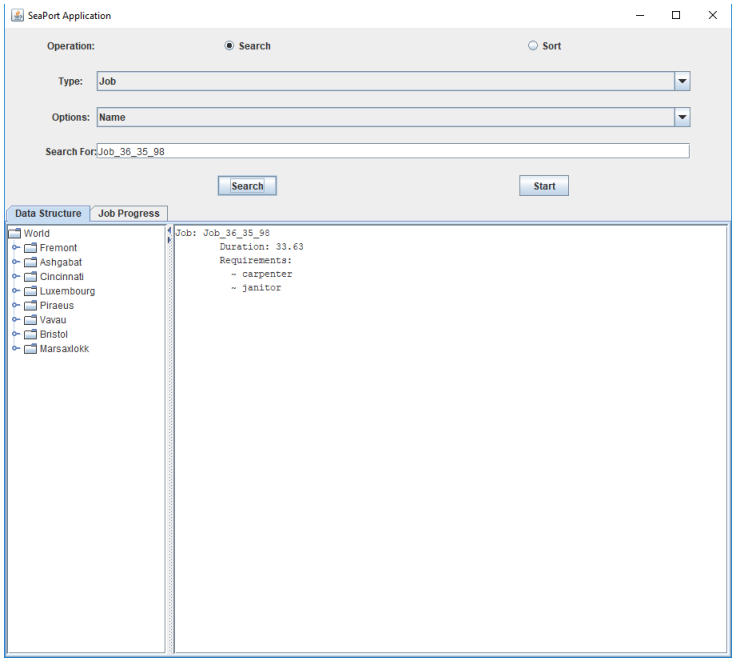


The program will display a confirmation of the file that has been selected.

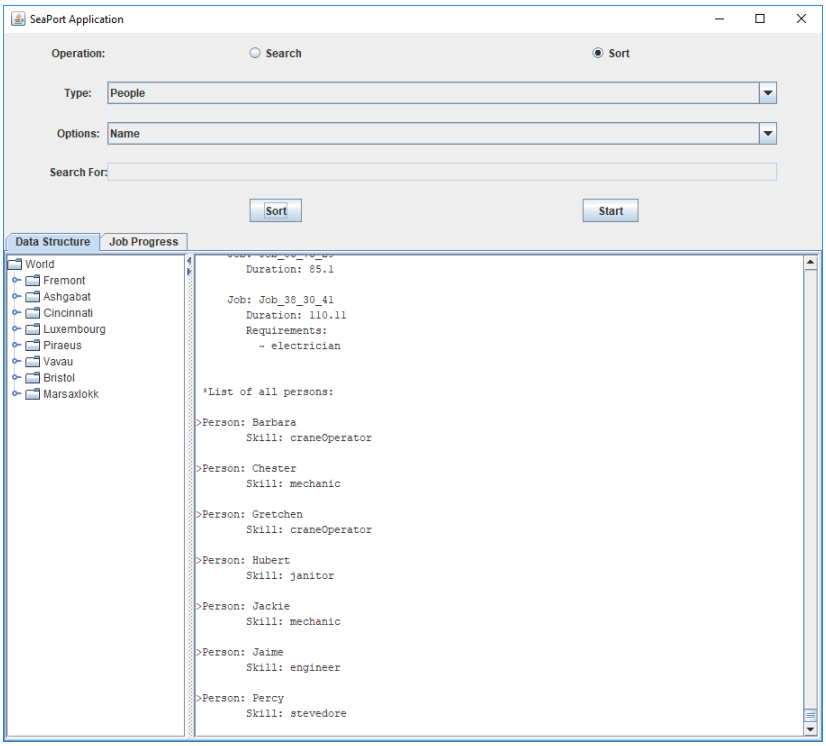


Once the text file is read, the data structure of the file will be displayed in a tree format under the first tab (labeled ‘Data Structures’) on the left side of the split pane. The information from the text file can now be searched or sorted. The operation to perform can be selected by clicking one of the buttons at the top of the GUI. The ‘Search’ operation allows you to search the file using various search criteria and the results will be printed in the text area on the right side of the split pane. Use the two drop-down menus under the input field to select the criteria you wish to search. Perform the search by clicking the ‘Search’ button. Alternatively, you can choose to sort information by selecting the ‘Sort’ button at the top. Options for sorting will be displayed in the drop-down menus. The results for this operation will be depicted in the tree on the left, as well as displayed in the text area on the right.

Search function: The results will be displayed on the right side of the split pane.



Sort function: The results will be displayed on both sides of the split pane – the tree will sort by the selected criteria and the sorted information will also be displayed on the right side.



The following fields can be searched and sorted using the SeaPortProgram4 GUI:

Search:

Person - Name, Skill

Job - Name, Requirement

Port - Name, Ship Name, Person Name

Docks - Name

Ships - Name

Sort:

Person - Name, Skill

Job - Name, Duration

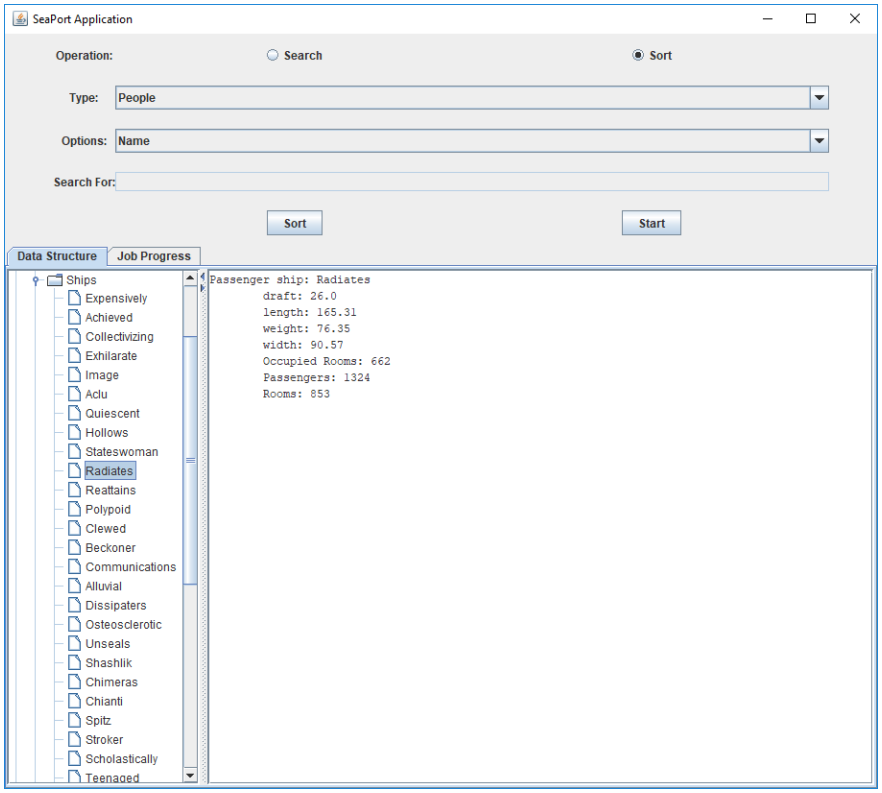
Port - Name

Docks - Name

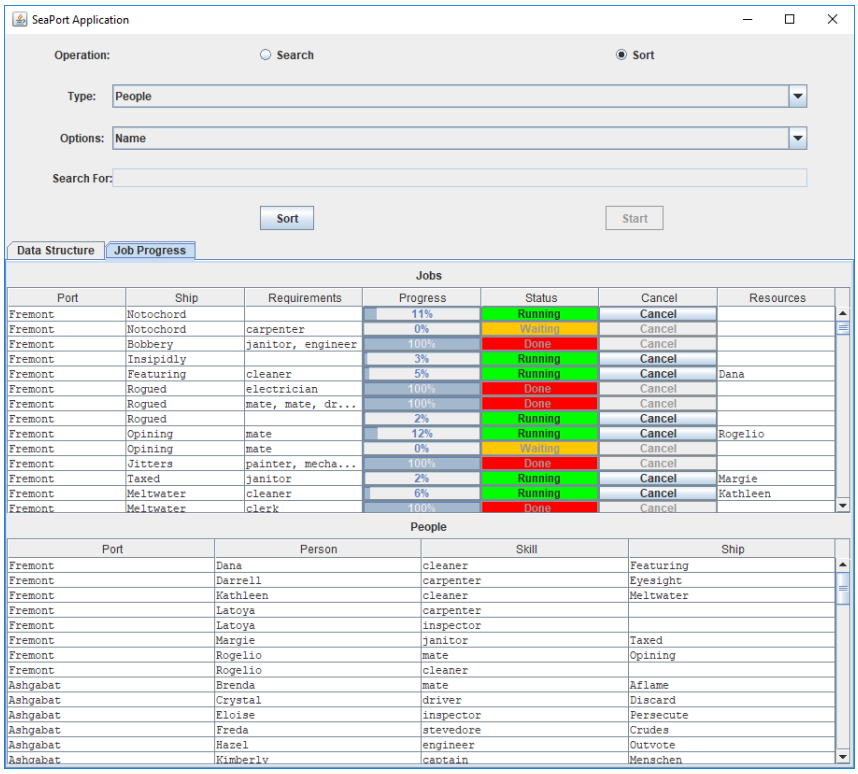
All Ships - Name, Draft, Length, Weight, Width

Ships in Que - Name, Draft, Length, Weight, Width

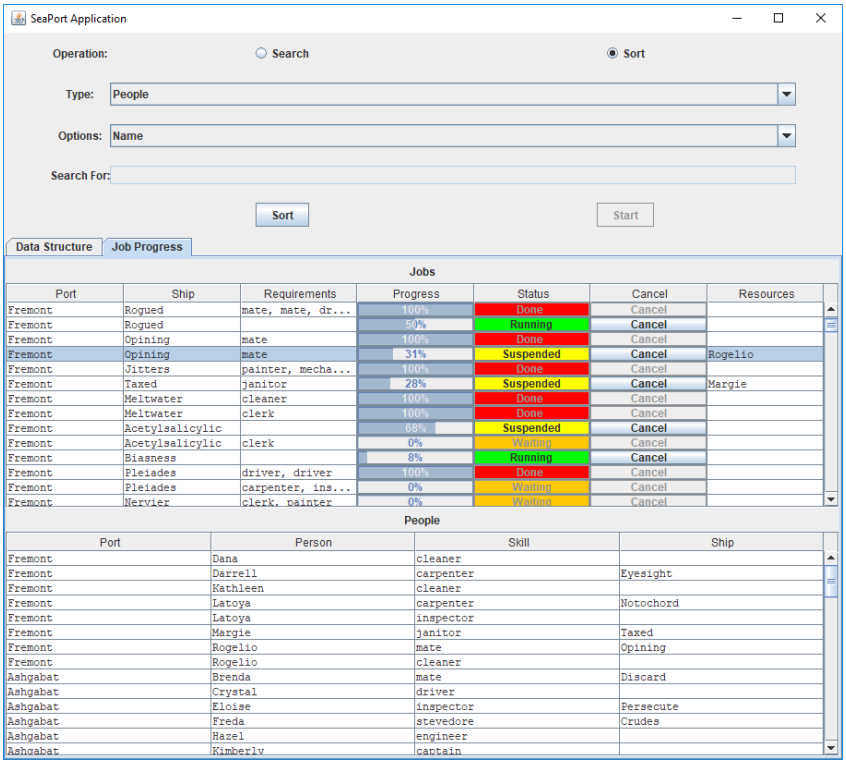
As described above, the data structure of the file is displayed in a tree format on the left side of the split pane. A unique feature of this tree is that when specific nodes are clicked, the information pertaining to that node is displayed on the right side of the split pane



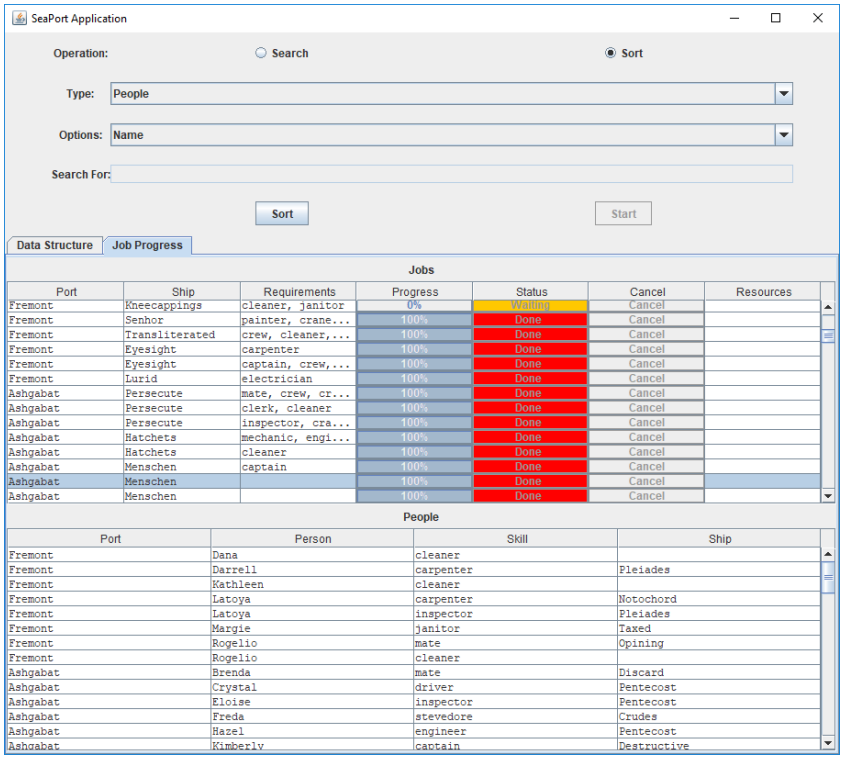
Finally, a threading process can be initiated for all jobs using the ‘Start’ button. The progress for the jobs is displayed under the tab labeled ‘Job Progress’. Two tables are displayed separated by a split pane. The top table (labeled ‘Jobs’) displays each job with its respective port, ship, requirements, progress, status, and resources (people) being utilized in order to run. The bottom table (labeled ‘People’) displays each person, along with the port they belong to, the skill they have, and the ship they are working on (when applicable). Any job that does not have the ability to acquire all resources (i.e. not all requirements can be fulfilled with the available people at a port) will cancel and its status will be displayed as “Done.”



Jobs can be suspended by clicking the ‘Status’ button next to a job when it says “Running.”



Jobs can also be cancelled for running jobs by clicking the ‘Cancel’ button.



Test Plan:

To understand this program, the following table is used for testing:

Test Case	Input	Test Subject / Expected Output	Actual Output	P/F
#1	*File chosen*: *aSPad.txt* Operation: N/A Type: N/A Options: N/A Search For: N/A Button Selected: N/A *Examining JTree on left side of JSplitPane and corresponding information printed on right side of JSplitPane under 'Data Structure' tab.	The data structure in the text file is displayed as a JTree on the left side of the JSplitPane under the 'Data Structure' tab. Docks, Ships, Ships in Que, and Persons are all displayed. Specific information pertaining to each node is displayed on the right side of the JSplitPane. This test case tests the creation of the JTree to show the data structure in the text file and the output of information next to the JTree. This case ensures that this functionality has been maintained from Project 3.	The data structure in the text file is displayed as a JTree on the left side of the JSplitPane under the 'Data Structure' tab. Docks, Ships, Ships in Que, and Persons are all displayed. Specific information pertaining to each node is displayed on the right side of the JSplitPane.	**P**
#2	*File chosen*: *aSPad.txt* Operation: Search Type: Ships Options: Name Search For: JUNCTURE Button Selected: Search	Passenger ship: Juncture draft: 31.71 length: 140.17 weight: 69.35 width: 45.19 Occupied Rooms: 84 Passengers: 168 Rooms: 840 This test case tests the Ships and Name search options of the GUI using all uppercase letters to ensure this	Passenger ship: Juncture draft: 31.71 length: 140.17 weight: 69.35 width: 45.19 Occupied Rooms: 84 Passengers: 168 Rooms: 840	**P**

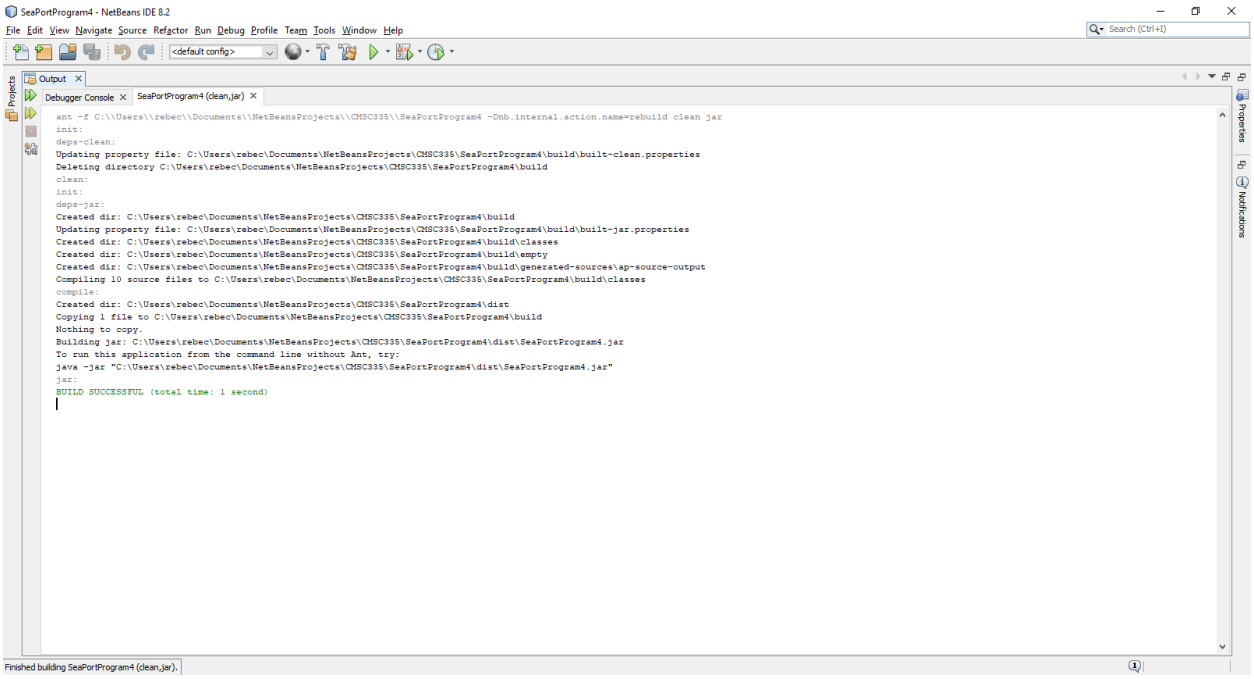
		functionality remains from Projects 1, 2, and 3.		
#3	*File chosen*: *aSPad.txt* Operation: Sort Type: Ports Options: Name Search For: N/A Button Selected: Sort	All ports sorted by name in World displayed on right side of the JSplitPane **and** sorted by name in the JTree on the left side of the JSplitPane. This test case tests the Ports and Name sort options of the GUI, an operation that was expanded on in Project 3. This case ensures that this functionality remains from Project 3.	All ports sorted by name in World displayed on right side of the JSplitPane **and** sorted by name in the JTree on the left side of the JSplitPane.	**P**
#4	*File chosen*: *aSPad.txt* Operation: Sort Type: Ships in Que Options: Width Search For: N/A Button Selected: Sort	All ships in que sorted by width in World displayed on right side of the JSplitPane **and** sorted by name in the JTree on the left side of the JSplitPane. This test case tests the Ships in Que and Width sort options of the GUI, an operation that was expanded on in Project 3. This case ensures that this functionality remains from Project 3.	All ships in que sorted by width in World displayed on right side of the JSplitPane **and** sorted by name in the JTree on the left side of the JSplitPane.	**P**
#5	*File chosen*: *aSPad.txt* Operation: N/A Type: N/A Options: N/A Search For: N/A Button Selected: Start	The threading process is initiated for all jobs and displayed in the 'Job Progress' tab of the JTabbedPane. This test case tests the multithreading feature of this program when the 'Start' button is pushed.	The threading process is initiated for all jobs and displayed in the 'Job Progress' tab of the JTabbedPane.	**P**

#6	*File chosen*: *aSPad.txt* Operation: N/A Type: N/A Options: N/A Search For: N/A Button Selected: N/A	The JTree will contain no ships listed as in a queue or at a dock. These nodes will be marked as *EMPTY*. This test case examines the content of the JTree *after* the jobs are run. This case ensures that this functionality remains from Project 3.	The JTree contain no ships listed as in a queue or at a dock. These nodes are marked as *EMPTY*.	**P**
#7	*File chosen*: *aSPad2.txt* Operation: N/A Type: N/A Options: N/A Search For: N/A Button Selected: Start o Click on 'Cancel' button for a few jobs: ▪ Job running at port Mersin for ship Quadrillionth ▪ Second job running at port Mersin for ship Tiptoed ▪ Job running at port Mersin for ship Acridly (using resource Cary)	All canceled jobs will display a status of 'Done', the resources (people) will be removed from the table, and they will be removed from the docks in which they were located in the JTree. This test case tests the functionality of the 'Cancel' button. When the 'Cancel' button is pushed for a job, the job is finished (because it is no longer needed). That job also releases the resources (people), that it used, if any, to progress. Once all jobs are done for that ship (including the canceled one(s)), the ship should still be removed from the dock at which it was located.	All canceled jobs display a status of 'Done', the resources (people) were removed from the table, and the jobs were removed from the docks in which they were located in the JTree.	**P**

#8	*File chosen: aSPad2.txt* Operation: Search Type: Dock Options: Name Search For: Pier_15 Button Selected: Search	Dock: Pier_15 >*EMPTY* This test case tests the Dock and Name search options of the GUI with upper case letters. Any dock searched should be empty since jobs have finished running. This case ensures that this functionality remains from Project 3.	Dock: Pier_15 >*EMPTY*	**P**
#9	*File chosen: aSPad2.txt* Operation: N/A Type: N/A Options: N/A Search For: N/A Button Selected: N/A *Examining Ships in Que node of JTree on left side of JSplitPane under 'Data Structure' tab	All of the Ships in Que nodes of the JTree will be marked as *EMPTY*. This test case tests the removal of ships from the Ships in Que. All ships listed under this node of the JTree should be removed once the jobs have finished running. This case ensures that this functionality remains from Project 3.	All of the Ships in Que nodes of the JTree are marked as *EMPTY*.	**P**
#10	*File chosen: aSPad3.txt* Operation: N/A Type: N/A Options: N/A Search For: N/A Button Selected: Start	All running jobs whose 'Status' button is clicked will be paused and the 'Status' button will turn yellow and be marked "Suspended." They will not release their resources (people). Also, they will remain in the dock, as depicted in the JTree. This test case tests the functionality of the 'Status'	All running jobs whose 'Status' button was clicked were paused; the 'Status' button turned yellow and was marked "Suspended." They did not release their resources (people). Also, these jobs remained in the dock, as depicted in the JTree. *Note: Two other jobs (i.e. for ships Stressors and*	**P**

	○ Click on 'Status' button when job is running (says "Running") for a few jobs: ▪ Job at port Linyi for ship Benched ▪ Job at port Linyi for ship Imperturbability (utilizing resources Johnnie and Mona) ▪ Job at port Linyi for ship Jackfishes (utilizing resources Wayne and Sue)	button. When the 'Status' button is pushed for a running job, the job is paused/suspended. Ships of suspended jobs should remain at the docks at which they are located. Resources (people) being utilized to run the job are not released. Any other jobs that have not been run for that ship remain in waiting.	*Aridly) are left in waiting because they need resources from the suspended jobs at port Linyi. They remain located at their respective docks.*	
#11	*<u>File chosen</u>: aSPad4.txt* <u>Operation</u>: N/A <u>Type</u>: N/A <u>Options</u>: N/A <u>Search For</u>: N/A <u>Button Selected</u>: Start *Examining the resources (people) in the 'Jobs' table to see that they are moving when jobs are completed and ensuring that their ship name changes in the 'People' table.	In the 'Jobs' table, resources (people) being used by running jobs will be moved once those jobs are complete to other jobs that were previously waiting to progress. Also, in the 'Jobs' table, the ship name where the person is working will change as the person switches jobs. This test case tests the functionality of the resources being utilized in the multithreading process to ensure that these resources (people) are being moved properly and that this is being displayed correctly in the tables in the 'Job Progress' tab.	In the 'Jobs' table, resources (people) being used by running jobs are moved once those jobs are complete to other jobs that were previously waiting to progress. Also, in the 'Jobs' table, the ship name where the person is working changes as the person switches jobs. *Note: The screenshots focus on the following resources (people): Cheryl, Frederick, Lester, Levi, and Diane. They move from ship to ship to work on jobs and this is depicted in both the 'Jobs' and 'People' tables.*	**P**

Compiling the source files (SeaPortProgram4.java, Thing.java, World.java, SeaPort.java, Dock.java, Ship.java, PassengerShip.java, CargoShip.java, Person.java, Job.java) *using -Xlint*—



Test Case #1—Input:

File chosen: *aSPad.txt*

Operation: N/A

Type: N/A

Options: N/A

Search For: N/A

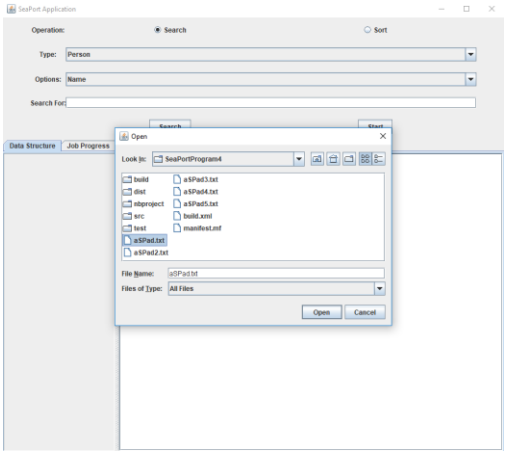
Button Selected: N/A

*Examining JTree on left side of JSplitPane and corresponding information printed on right side of JSplitPane under 'Data Structure' tab.

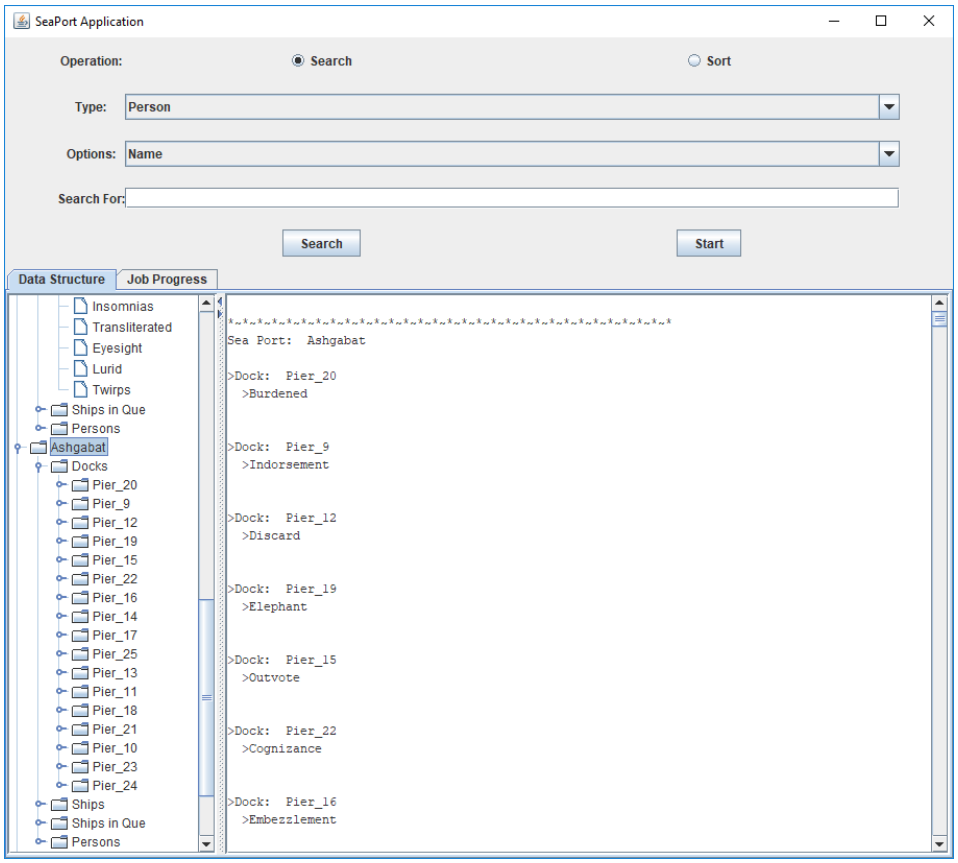
This test case tests the creation of the JTree to show the data structure in the text file and the output of information next to the JTree. This case ensures that this functionality has been maintained from Project 3.

The expected results for this case are as follows:

The data structure in the text file is displayed as a JTree on the left side of the JSplitPane under the 'Data Structure' tab. Docks, Ships, Ships in Que, and Persons are all displayed. Specific information pertaining to each node is displayed on the right side of the JSplitPane.



<u>Test Case #1 – Output</u>: Actual results match the expected output in our test table.



Test Case #2—Input:

File chosen: *aSPad.txt*

Operation: Search

Type: Ships

Options: Name

Search For: JUNCTURE

Button Selected: Search

This test case tests the Ships and Name search options of the GUI using all uppercase letters to ensure this functionality remains from Projects 1, 2, and 3.

The expected results for this case are as follows:

Passenger ship: Juncture

draft: 31.71

length: 140.17

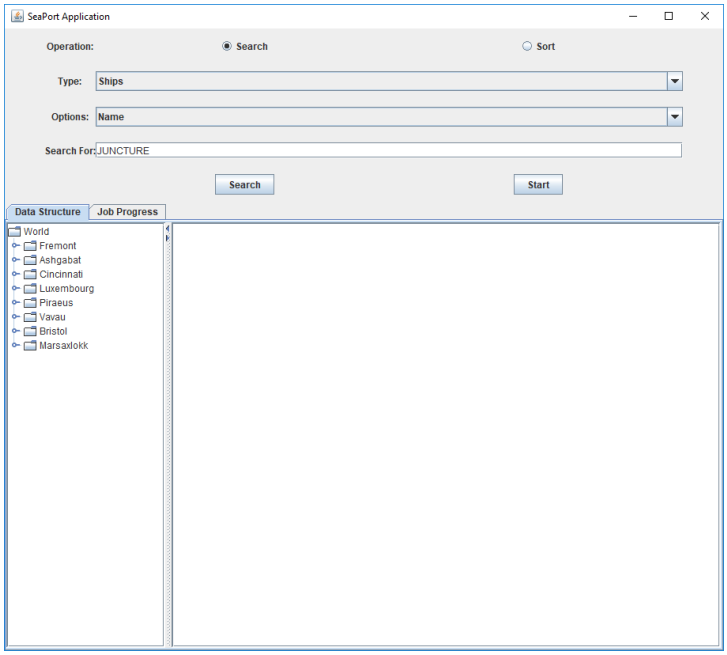
weight: 69.35

width: 45.19

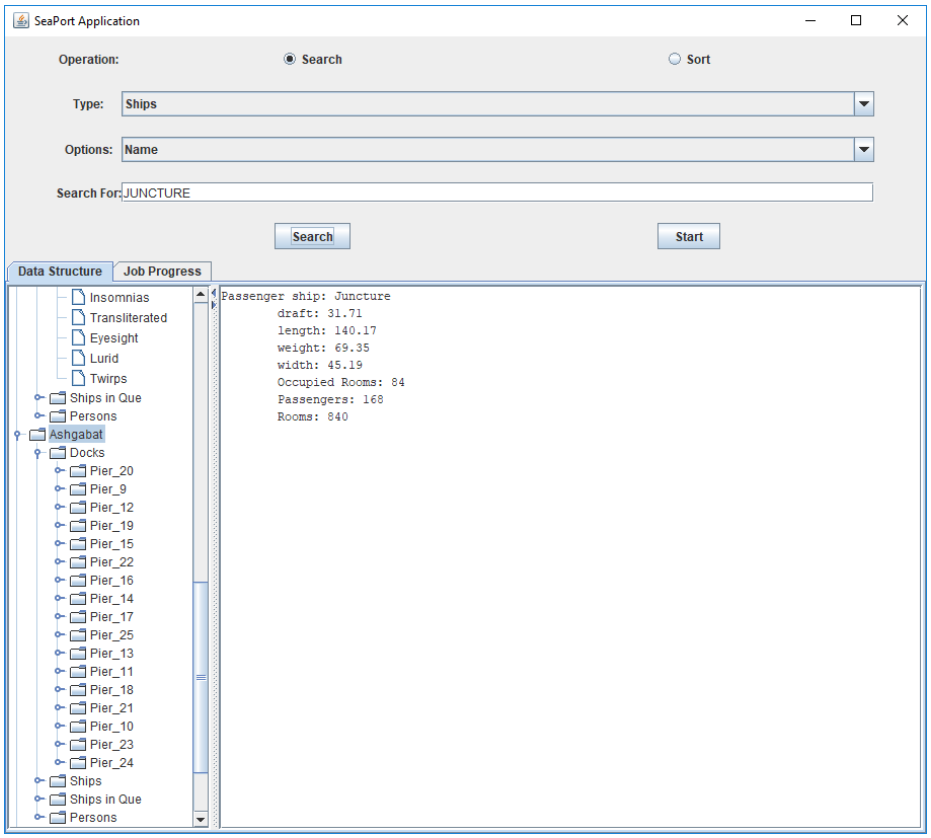
Occupied Rooms: 84

Passengers: 168

Rooms: 840



<u>Test Case #2 – Output</u>: Actual results match the expected output in our test table.



Test Case #3—Input:

File chosen: aSPad.txt

Operation: Sort

Type: Ports

Options: Name

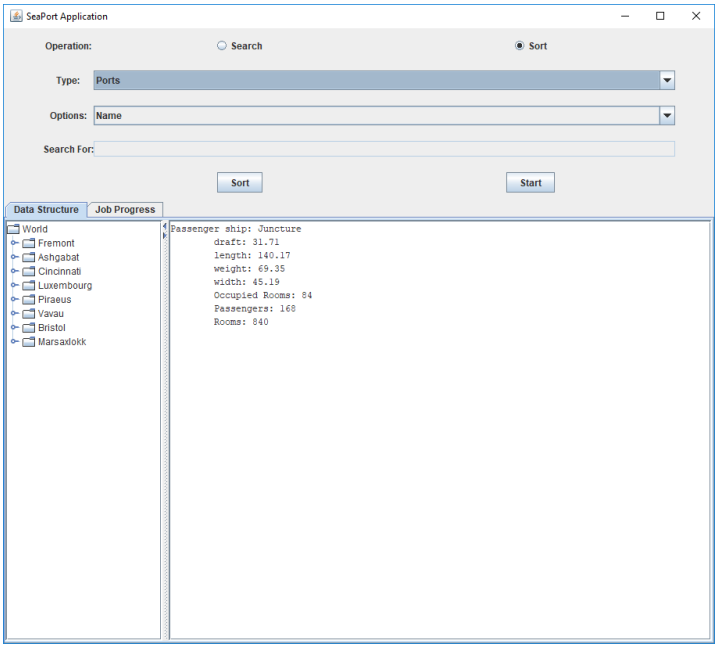
Search For: N/A

Button Selected: Sort

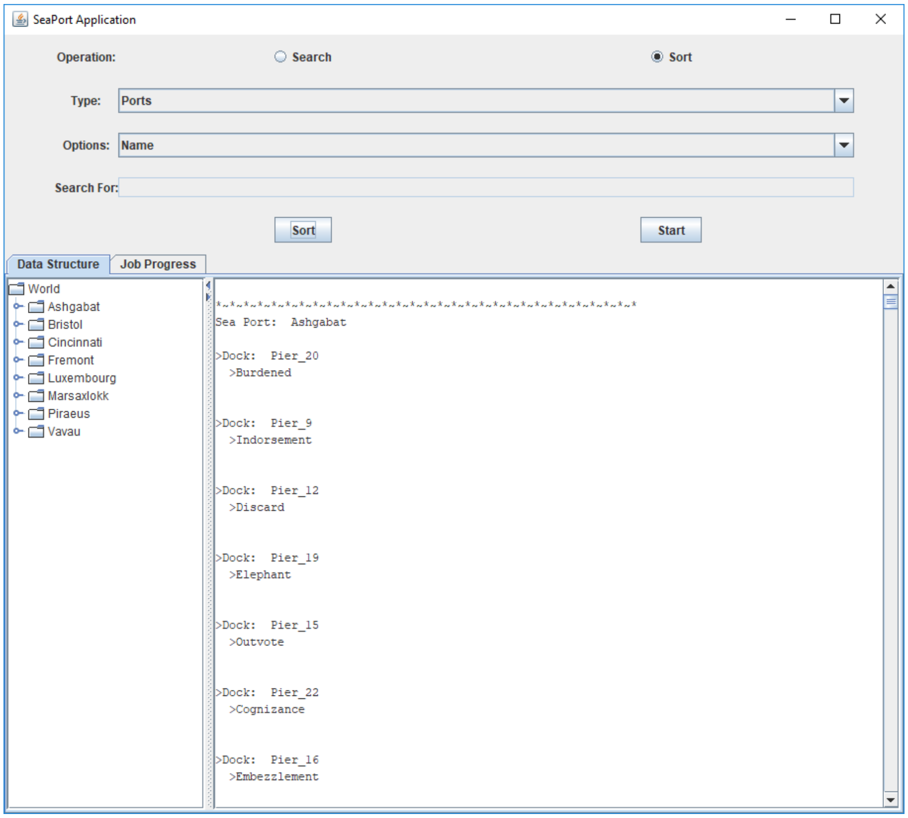
This test case tests the Ports and Name sort options of the GUI, an operation that was expanded on in Project 3. This case ensures that this functionality remains from Project 3.

The expected results for this case are as follows:

All ports sorted by name in World displayed on right side of the JSplitPane **and** sorted by name in the JTree on the left side of the JSplitPane.



<u>Test Case #3 – Output</u>: Actual results match the expected output in our test table.



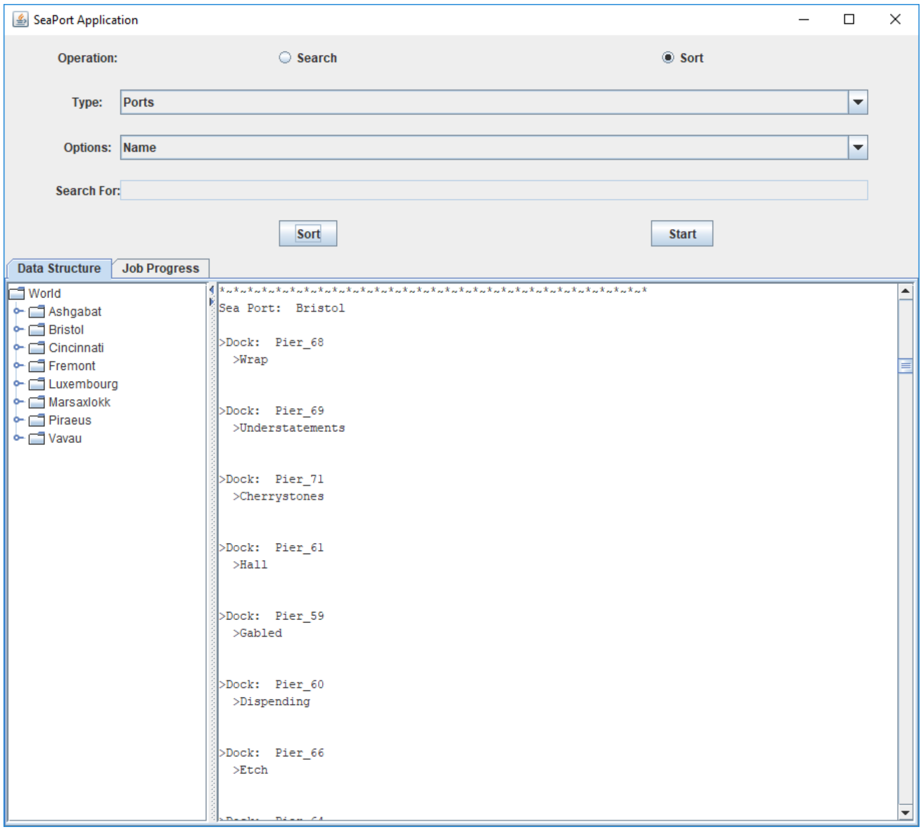
Operation: ● Sort

Type: Ports

Options: Name

Search For:

```
*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*
Sea Port:  Ashgabat

>Dock:  Pier_20
  >Burdened


>Dock:  Pier_9
  >Indorsement


>Dock:  Pier_12
  >Discard


>Dock:  Pier_19
  >Elephant


>Dock:  Pier_15
  >Outvote


>Dock:  Pier_22
  >Cognizance


>Dock:  Pier_16
  >Embezzlement
```



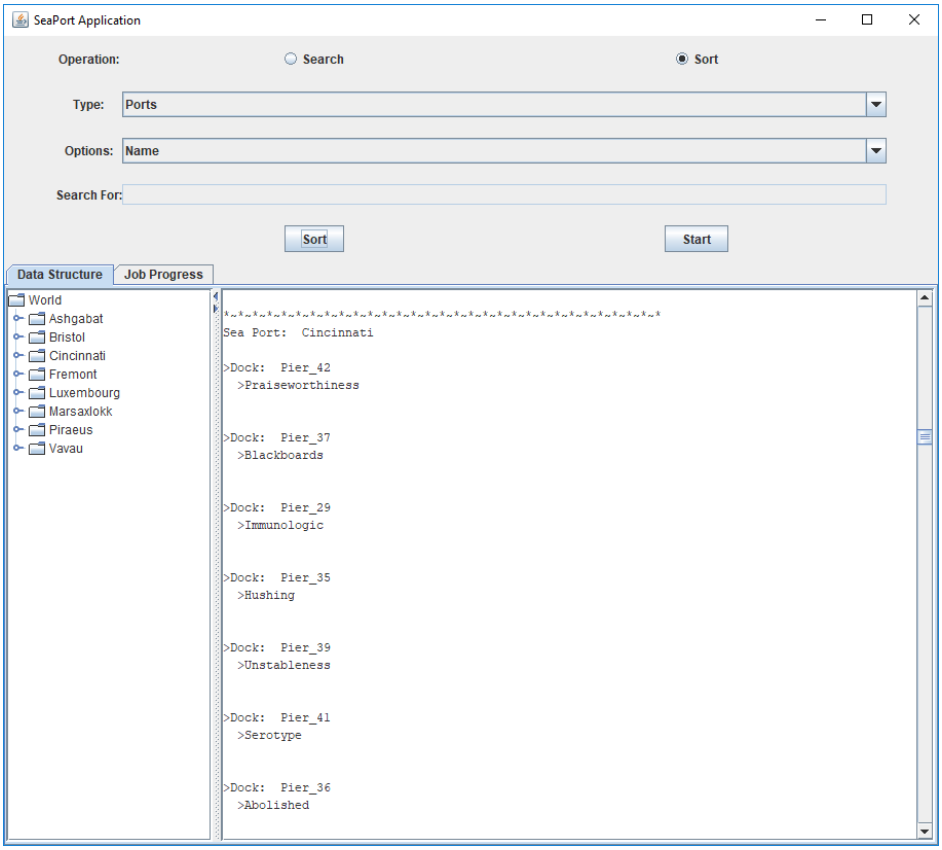
Operation: ● Sort

Type: Ports

Options: Name

Search For:

```
*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*
Sea Port:  Bristol

>Dock:  Pier_68
  >Wrap


>Dock:  Pier_69
  >Understatements


>Dock:  Pier_71
  >Cherrystones


>Dock:  Pier_61
  >Hall


>Dock:  Pier_59
  >Gabled


>Dock:  Pier_60
  >Dispending


>Dock:  Pier_66
  >Etch
```


SeaPort Application
Operation:
Search
Sort
Type:
Ports
Options:
Name
Search For:
Sort
Start
Data Structure
Job Progress
World
Ashgabat
Bristol
Cincinnati
Fremont
Luxembourg
Marsaxlokk
Piraeus
Vavau
~
Sea Port: Cincinnati
>Dock: Pier_42
>Praiseworthiness
>Dock: Pier_37
>Blackboards
>Dock: Pier_29
>Immunologic
>Dock: Pier_35
>Hushing
>Dock: Pier_39
>Unstableness
>Dock: Pier_41
>Serotype
>Dock: Pier_36
>Abolished

Test Case #4—Input:

File chosen: aSPad.txt

Operation: Sort

Type: Ships in Que

Options: Width

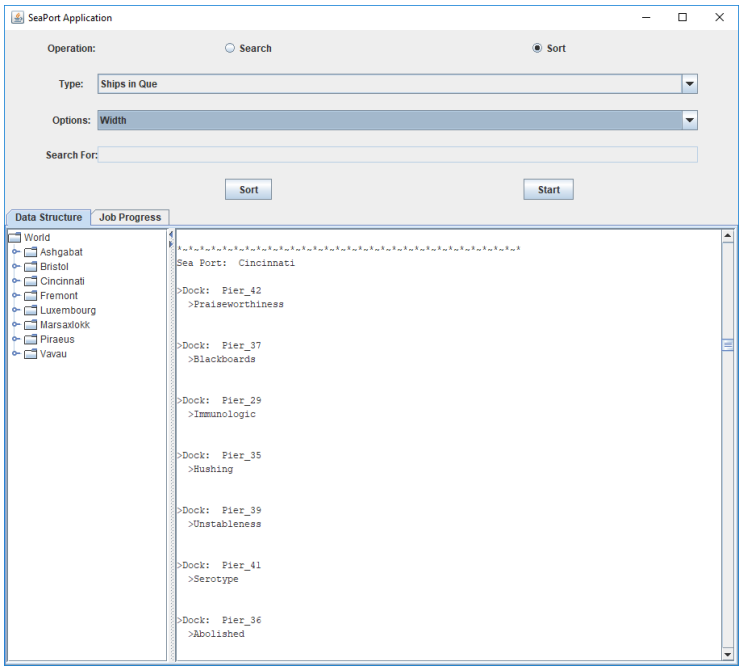
Search For: N/A

Button Selected: Sort

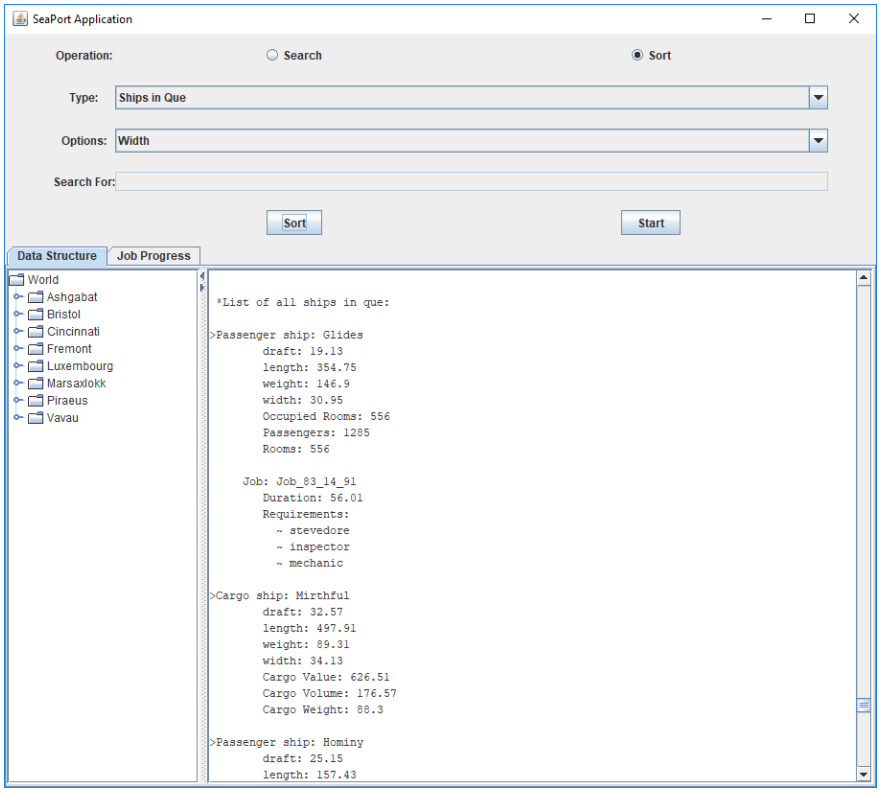
This test case tests the Ships in Que and Width sort options of the GUI, an operation that was expanded on in Project 3. This case ensures that this functionality remains from Project 3.

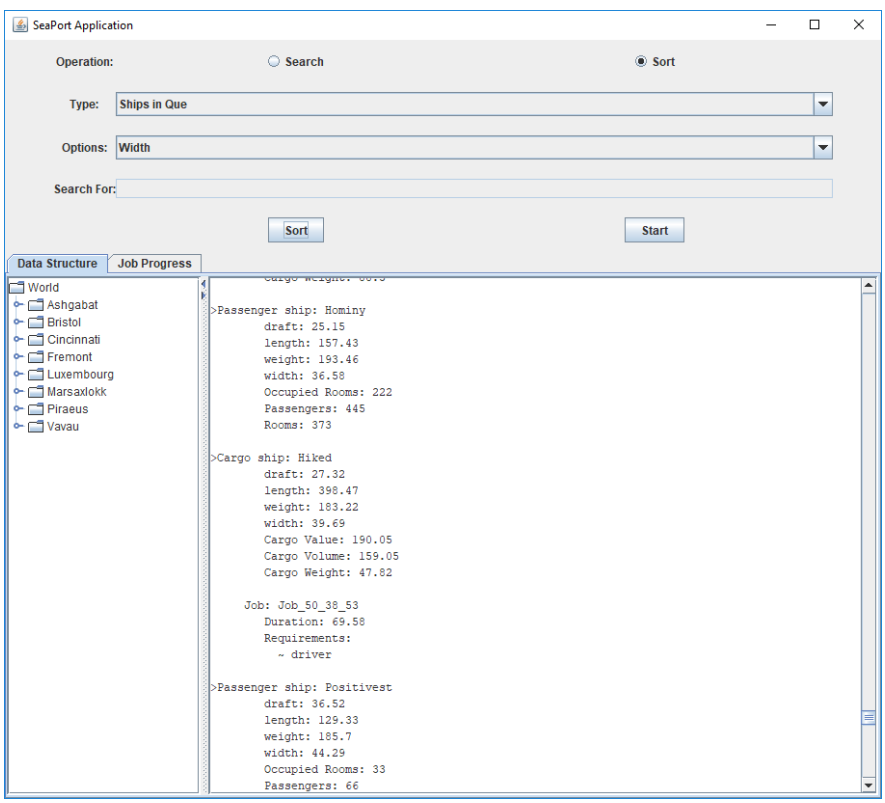
The expected results for this case are as follows:

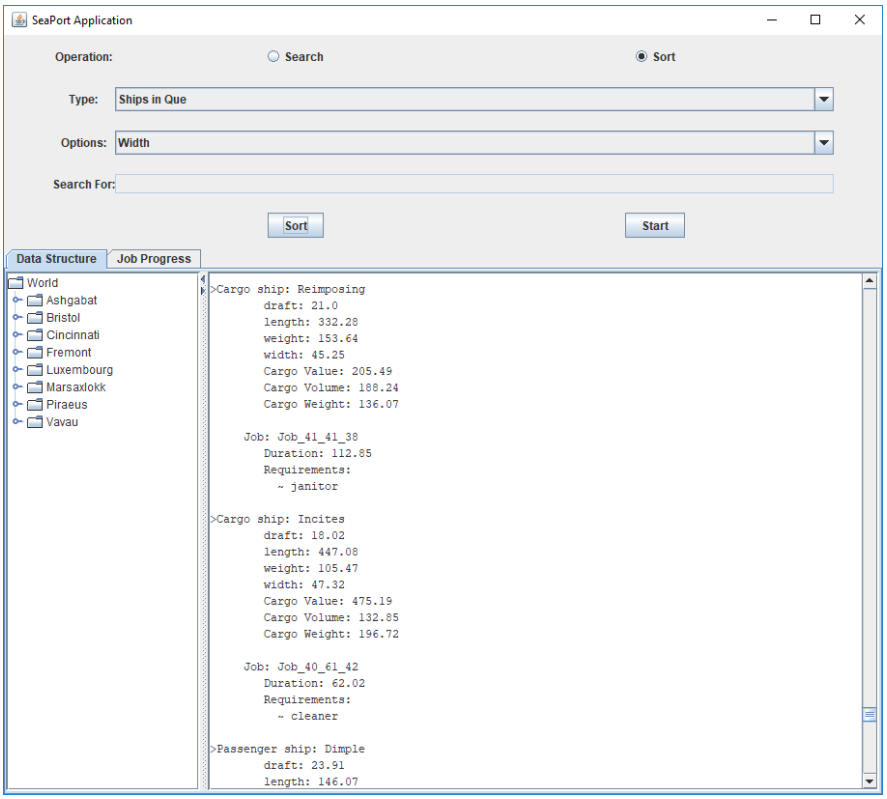
All ships in que sorted by width in World displayed on right side of the JSplitPane **and** sorted by name in the JTree on the left side of the JSplitPane.



<u>Test Case #4 – Output</u>: Actual results match the expected output in our test table.






SeaPort Application
Operation:
Search
Sort
Type:
Ships in Que
Options:
Width
Search For:
Sort
Start
Data Structure
Job Progress
World
Ashgabat
Bristol
Cincinnati
Fremont
Luxembourg
Marsaxlokk
Piraeus
Vavau
>Cargo ship: Reimposing
draft: 21.0
length: 332.28
weight: 153.64
width: 45.25
Cargo Value: 205.49
Cargo Volume: 188.24
Cargo Weight: 136.07
Job: Job_41_41_38
Duration: 112.85
Requirements:
~ janitor
>Cargo ship: Incites
draft: 18.02
length: 447.08
weight: 105.47
width: 47.32
Cargo Value: 475.19
Cargo Volume: 132.85
Cargo Weight: 196.72
Job: Job_40_61_42
Duration: 62.02
Requirements:
~ cleaner
>Passenger ship: Dimple
draft: 23.91
length: 146.07

Test Case #5—Input:

File chosen: aSPad.txt

Operation: N/A

Type: N/A

Options: N/A

Search For: N/A

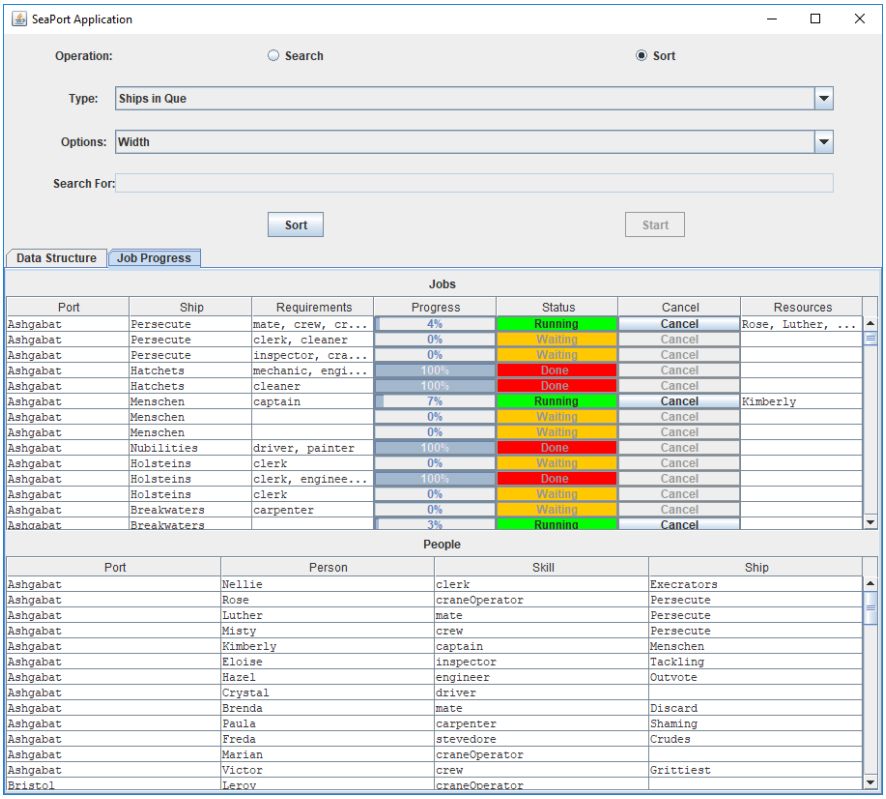
Button Selected: Start

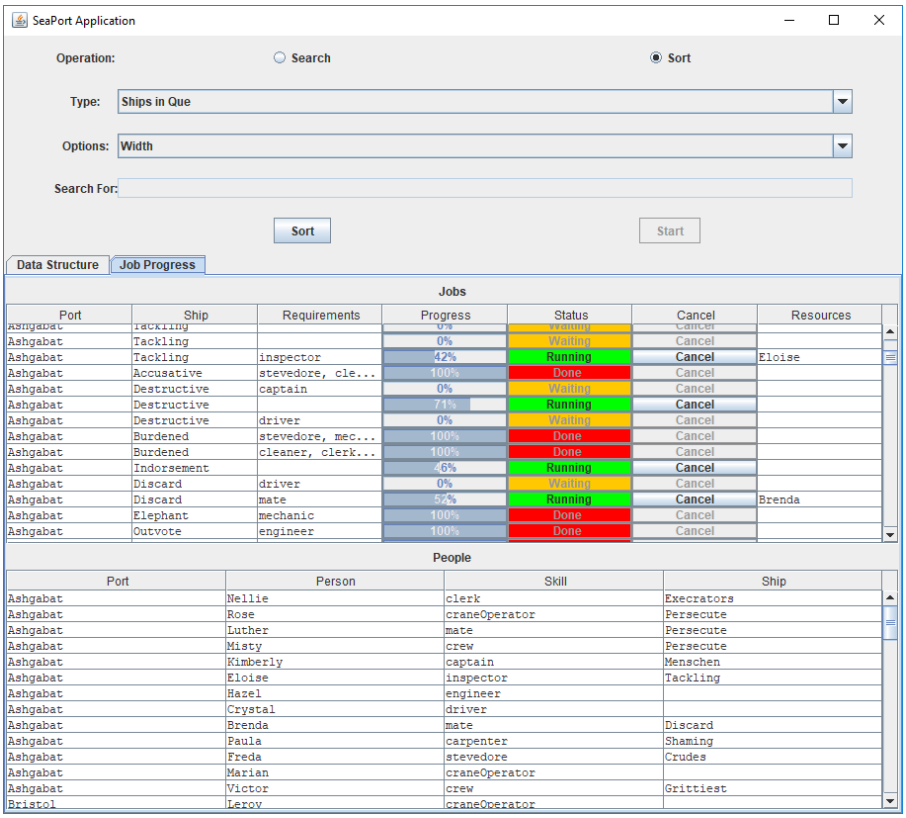
This test case tests the multithreading feature of this program when the 'Start' button is pushed.

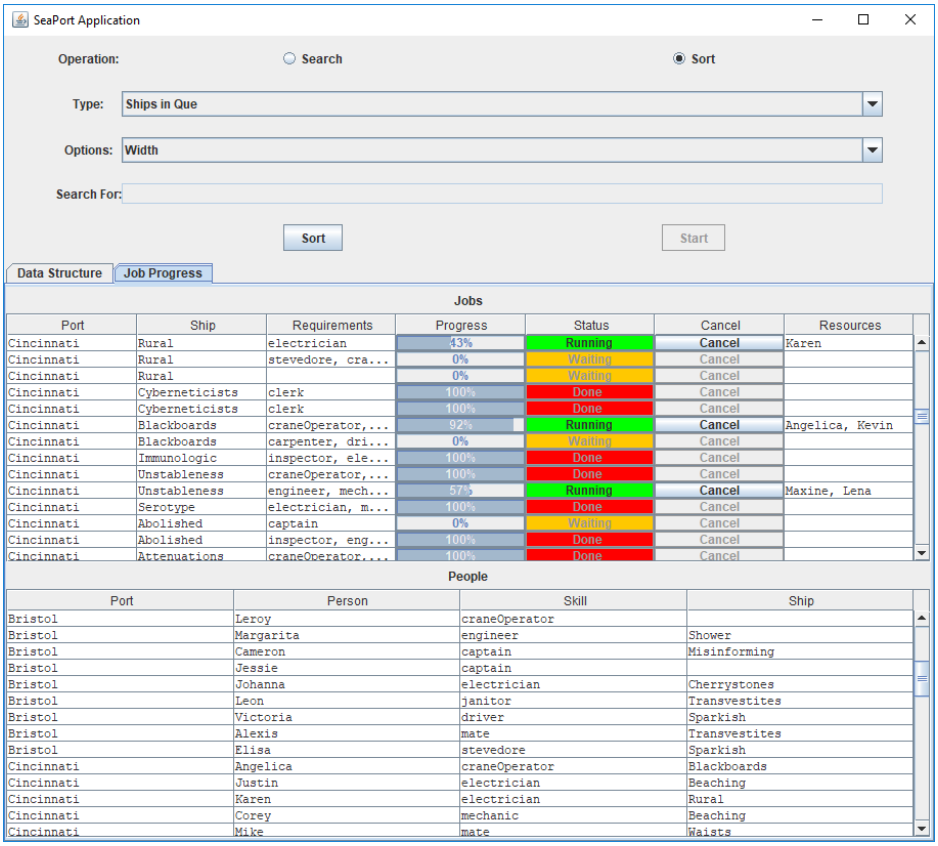
The expected results for this case are as follows:

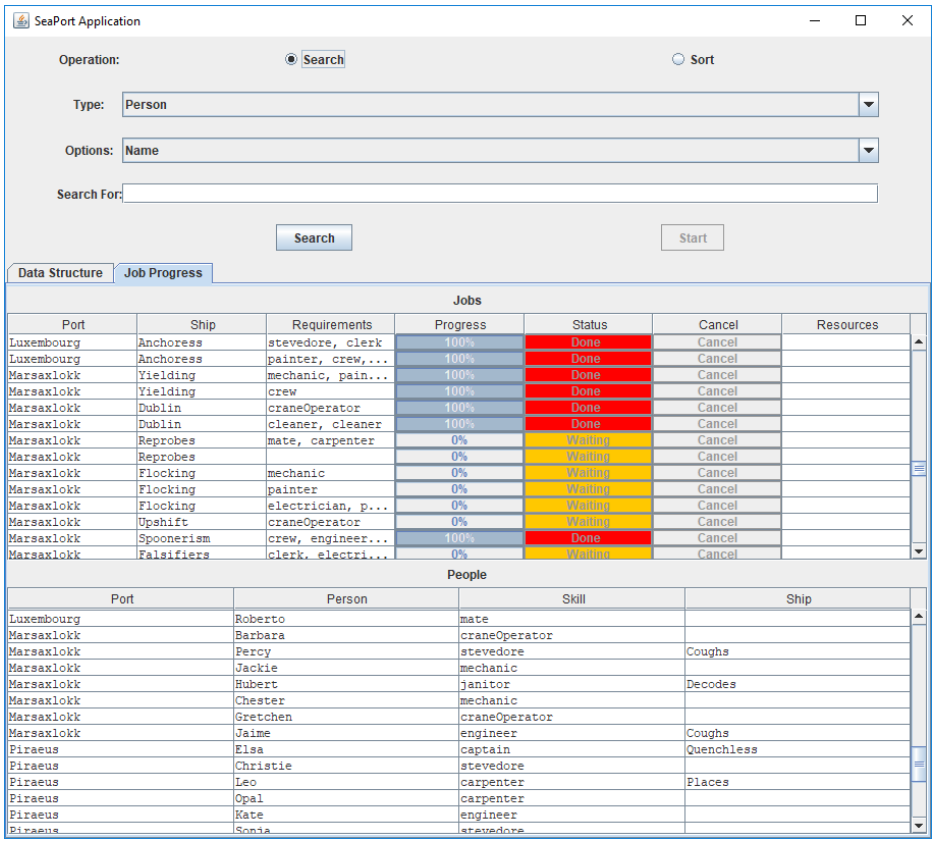
The threading process is initiated for all jobs and displayed in the 'Job Progress' tab of the JTabbedPane.

<u>Test Case #5 – Output</u>: Actual results match the expected output in our test table.






SeaPort Application
Operation:
Search
Sort
Type:
Ships in Que
Options:
Width
Search For:
Sort
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Cincinnati | Rural | electrician | 43% | Running | Cancel | Karen
Cincinnati | Rural | stevedore, cra... | 0% | Waiting | Cancel
Cincinnati | Rural | | 0% | Waiting | Cancel
Cincinnati | Cyberneticists | clerk | 100% | Done | Cancel
Cincinnati | Cyberneticists | clerk | 100% | Done | Cancel
Cincinnati | Blackboards | craneOperator,... | 92% | Running | Cancel | Angelica, Kevin
Cincinnati | Blackboards | carpenter, dri... | 0% | Waiting | Cancel
Cincinnati | Immunologic | inspector, ele... | 100% | Done | Cancel
Cincinnati | Unstableness | craneOperator,... | 100% | Done | Cancel
Cincinnati | Unstableness | engineer, mech... | 57% | Running | Cancel | Maxine, Lena
Cincinnati | Serotype | electrician, m... | 100% | Done | Cancel
Cincinnati | Abolished | captain | 0% | Waiting | Cancel
Cincinnati | Abolished | inspector, eng... | 100% | Done | Cancel
Cincinnati | Attenuations | craneOperator,... | 100% | Done | Cancel
People
Port | Person | Skill | Ship
Bristol | Leroy | craneOperator |
Bristol | Margarita | engineer | Shower
Bristol | Cameron | captain | Misinforming
Bristol | Jessie | captain |
Bristol | Johanna | electrician | Cherrystones
Bristol | Leon | janitor | Transvestites
Bristol | Victoria | driver | Sparkish
Bristol | Alexis | mate | Transvestites
Bristol | Elisa | stevedore | Sparkish
Cincinnati | Angelica | craneOperator | Blackboards
Cincinnati | Justin | electrician | Beaching
Cincinnati | Karen | electrician | Rural
Cincinnati | Corey | mechanic | Beaching
Cincinnati | Mike | mate | Waists


SeaPort Application
Operation:
Search
Sort
Type:
Person
Options:
Name
Search For:
Search
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Luxembourg | Anchoress | stevedore, clerk | 100% | Done | Cancel
Luxembourg | Anchoress | painter, crew,... | 100% | Done | Cancel
Marsaxlokk | Yielding | mechanic, pain... | 100% | Done | Cancel
Marsaxlokk | Yielding | crew | 100% | Done | Cancel
Marsaxlokk | Dublin | craneOperator | 100% | Done | Cancel
Marsaxlokk | Dublin | cleaner, cleaner | 100% | Done | Cancel
Marsaxlokk | Reprobes | mate, carpenter | 0% | Waiting | Cancel
Marsaxlokk | Reprobes | | 0% | Waiting | Cancel
Marsaxlokk | Flocking | mechanic | 0% | Waiting | Cancel
Marsaxlokk | Flocking | painter | 0% | Waiting | Cancel
Marsaxlokk | Flocking | electrician, p... | 0% | Waiting | Cancel
Marsaxlokk | Upshift | craneOperator | 0% | Waiting | Cancel
Marsaxlokk | Spoonerism | crew, engineer... | 100% | Done | Cancel
Marsaxlokk | Falsifiers | clerk, electri... | 0% | Waiting | Cancel
People
Port | Person | Skill | Ship
Luxembourg | Roberto | mate |
Marsaxlokk | Barbara | craneOperator |
Marsaxlokk | Percy | stevedore | Coughs
Marsaxlokk | Jackie | mechanic |
Marsaxlokk | Hubert | janitor | Decodes
Marsaxlokk | Chester | mechanic |
Marsaxlokk | Gretchen | craneOperator |
Marsaxlokk | Jaime | engineer | Coughs
Piraeus | Elsa | captain | Quenchless
Piraeus | Christie | stevedore |
Piraeus | Leo | carpenter | Places
Piraeus | Opal | carpenter |
Piraeus | Kate | engineer |
Piraeus | Sonja | stevedore |

Test Case #6—Input:

File chosen: aSPad.txt

Operation: N/A

Type: N/A

Options: N/A

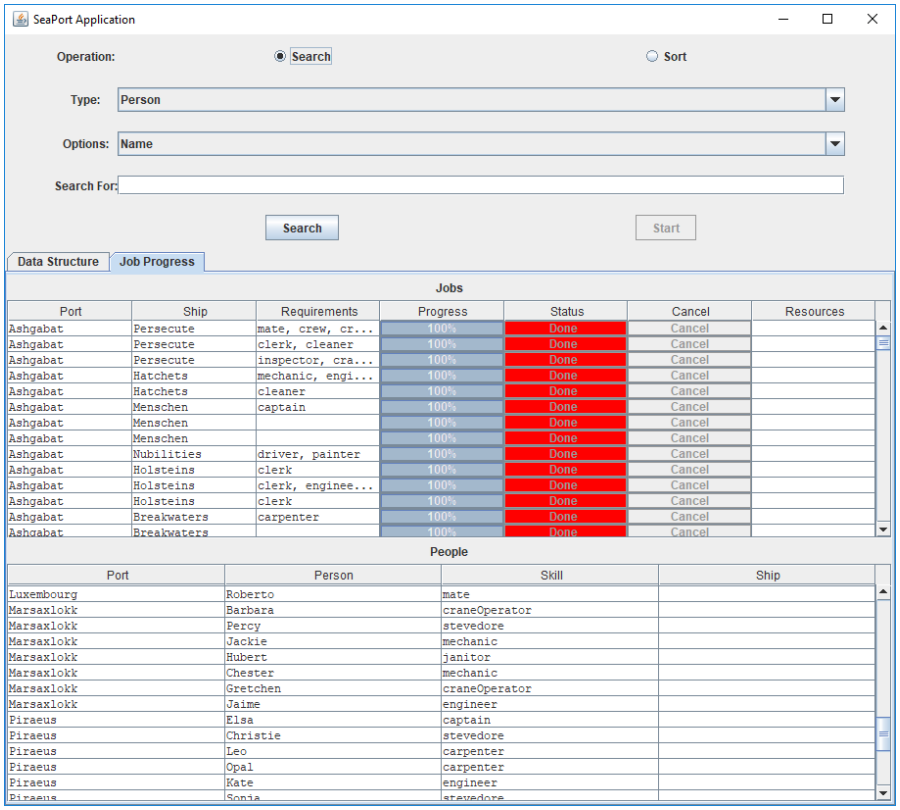
Search For: N/A

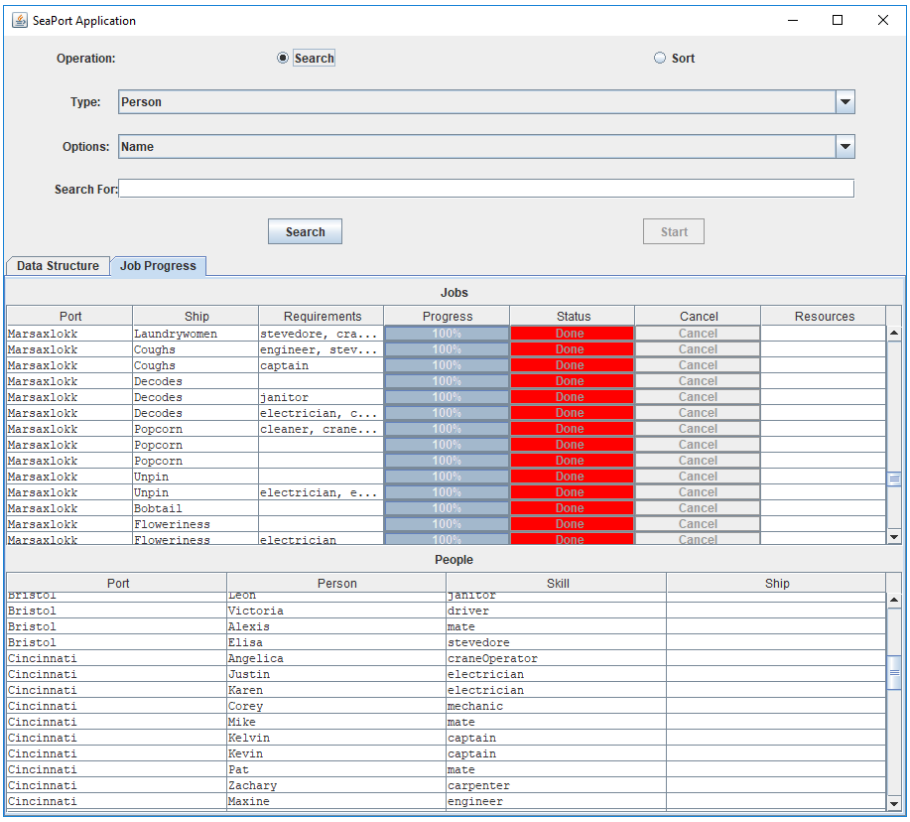
Button Selected: N/A

This test case examines the content of the JTree *after* the jobs are run. This case ensures that this functionality remains from Project 3.

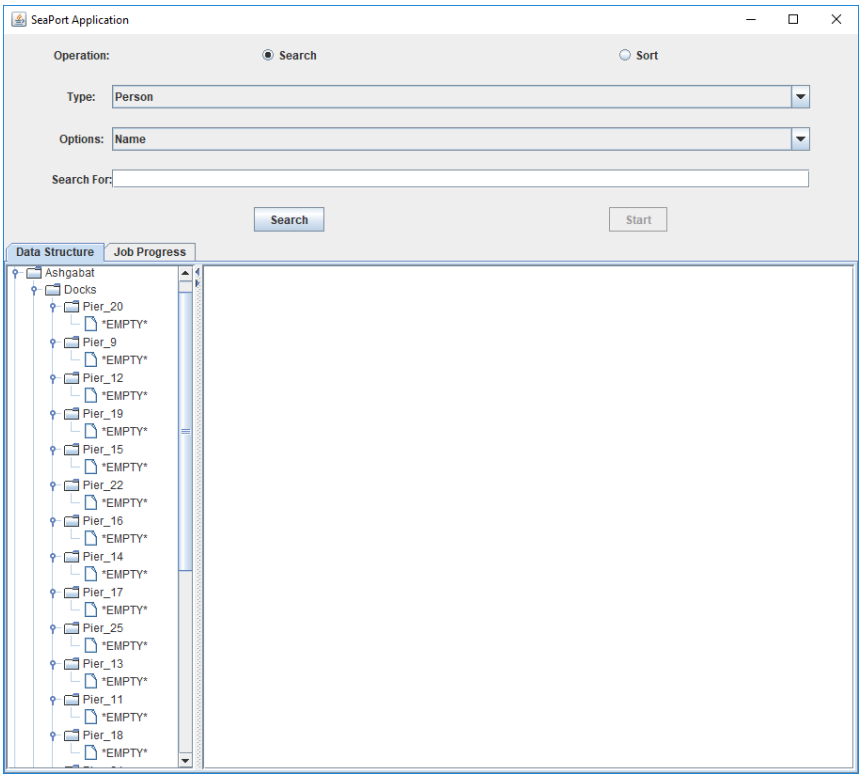
The expected results for this case are as follows:

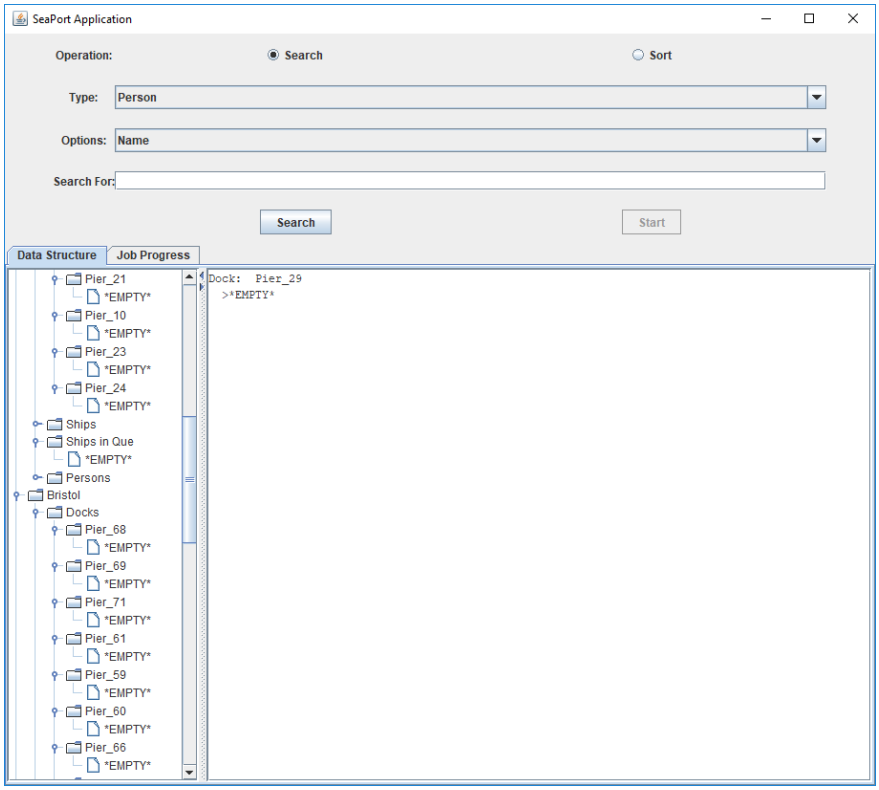
The JTree will contain no ships listed as in a queue or at a dock. These nodes will be marked as *EMPTY*.

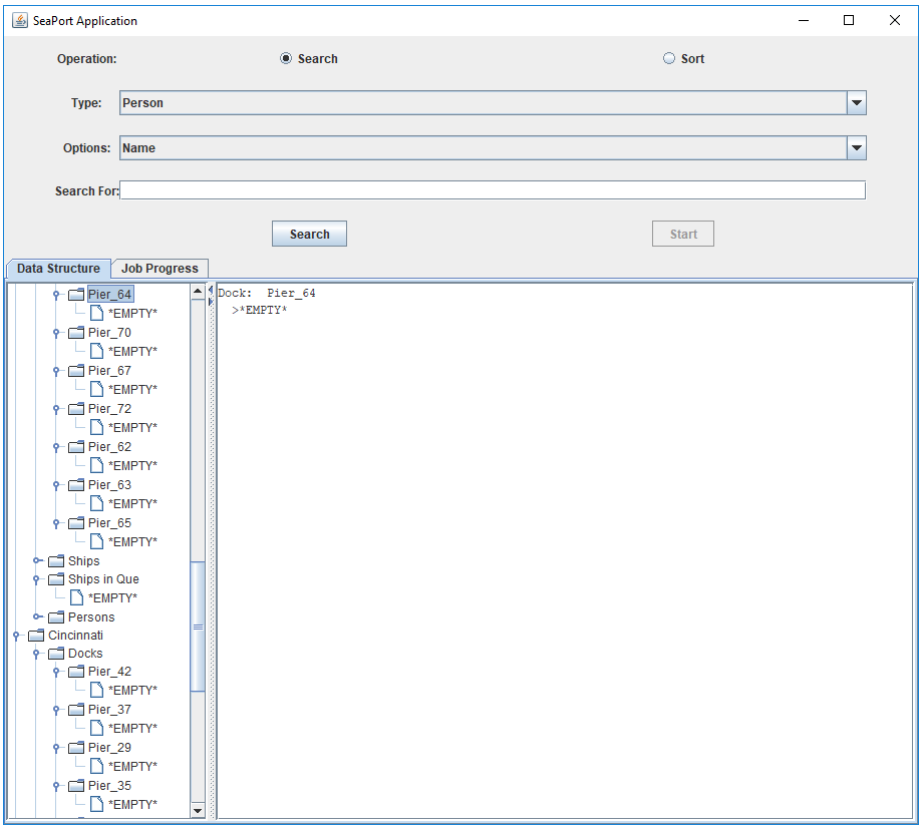

SeaPort Application
Operation:
Search
Sort
Type:
Person
Options:
Name
Search For:
Search
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Ashgabat | Persecute | mate, crew, cr... | 100% | Done | Cancel
Ashgabat | Persecute | clerk, cleaner | 100% | Done | Cancel
Ashgabat | Persecute | inspector, cra... | 100% | Done | Cancel
Ashgabat | Hatchets | mechanic, engi... | 100% | Done | Cancel
Ashgabat | Hatchets | cleaner | 100% | Done | Cancel
Ashgabat | Menschen | captain | 100% | Done | Cancel
Ashgabat | Menschen | | 100% | Done | Cancel
Ashgabat | Menschen | | 100% | Done | Cancel
Ashgabat | Nubilities | driver, painter | 100% | Done | Cancel
Ashgabat | Holsteins | clerk | 100% | Done | Cancel
Ashgabat | Holsteins | clerk, enginee... | 100% | Done | Cancel
Ashgabat | Holsteins | clerk | 100% | Done | Cancel
Ashgabat | Breakwaters | carpenter | 100% | Done | Cancel
Ashgabat | Breakwaters | | 100% | Done | Cancel
People
Port | Person | Skill | Ship
Luxembourg | Roberto | mate
Marsaxlokk | Barbara | craneOperator
Marsaxlokk | Percy | stevedore
Marsaxlokk | Jackie | mechanic
Marsaxlokk | Hubert | janitor
Marsaxlokk | Chester | mechanic
Marsaxlokk | Gretchen | craneOperator
Marsaxlokk | Jaime | engineer
Piraeus | Elsa | captain
Piraeus | Christie | stevedore
Piraeus | Leo | carpenter
Piraeus | Opal | carpenter
Piraeus | Kate | engineer


SeaPort Application
Operation:
Search
Sort
Type:
Person
Options:
Name
Search For:
Search
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Marsaxlokk | Laundrywomen | stevedore, cra... | 100% | Done | Cancel
Marsaxlokk | Coughs | engineer, stev... | 100% | Done | Cancel
Marsaxlokk | Coughs | captain | 100% | Done | Cancel
Marsaxlokk | Decodes | | 100% | Done | Cancel
Marsaxlokk | Decodes | janitor | 100% | Done | Cancel
Marsaxlokk | Decodes | electrician, c... | 100% | Done | Cancel
Marsaxlokk | Popcorn | cleaner, crane... | 100% | Done | Cancel
Marsaxlokk | Popcorn | | 100% | Done | Cancel
Marsaxlokk | Popcorn | | 100% | Done | Cancel
Marsaxlokk | Unpin | | 100% | Done | Cancel
Marsaxlokk | Unpin | electrician, e... | 100% | Done | Cancel
Marsaxlokk | Bobtail | | 100% | Done | Cancel
Marsaxlokk | Floweriness | | 100% | Done | Cancel
Marsaxlokk | Floweriness | electrician | 100% | Done | Cancel
People
Port | Person | Skill | Ship
Bristol | Leon | janitor
Bristol | Victoria | driver
Bristol | Alexis | mate
Bristol | Elisa | stevedore
Cincinnati | Angelica | craneOperator
Cincinnati | Justin | electrician
Cincinnati | Karen | electrician
Cincinnati | Corey | mechanic
Cincinnati | Mike | mate
Cincinnati | Kelvin | captain
Cincinnati | Kevin | captain
Cincinnati | Pat | mate
Cincinnati | Zachary | carpenter
Cincinnati | Maxine | engineer

Test Case #6—Output: Actual results match the expected output in our test table.






SeaPort Application
Operation:
Search
Sort
Type:
Person
Options:
Name
Search For:
Search
Start
Data Structure
Job Progress
Pier_64
EMPTY
Pier_70
EMPTY
Pier_67
EMPTY
Pier_72
EMPTY
Pier_62
EMPTY
Pier_63
EMPTY
Pier_65
EMPTY
Ships
Ships in Que
EMPTY
Persons
Cincinnati
Docks
Pier_42
EMPTY
Pier_37
EMPTY
Pier_29
EMPTY
Pier_35
EMPTY
Dock: Pier_64
>*EMPTY*

Test Case #7—Input:

File chosen: aSPad2.txt

Operation: N/A

Type: N/A

Options: N/A

Search For: N/A

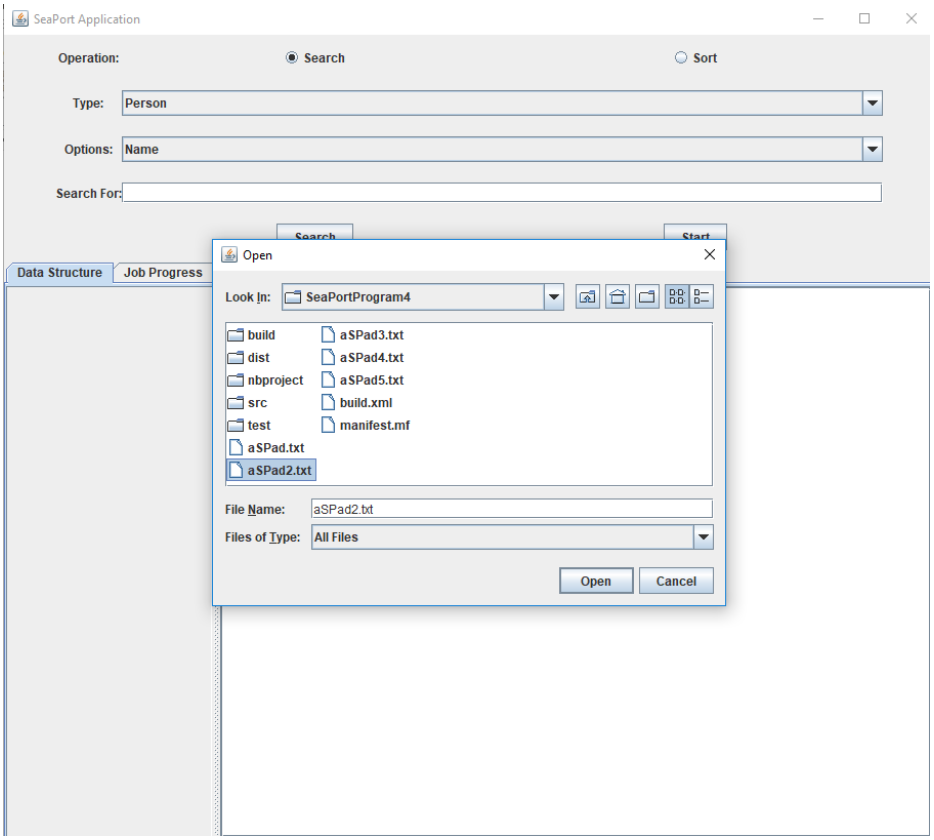
Button Selected: Start

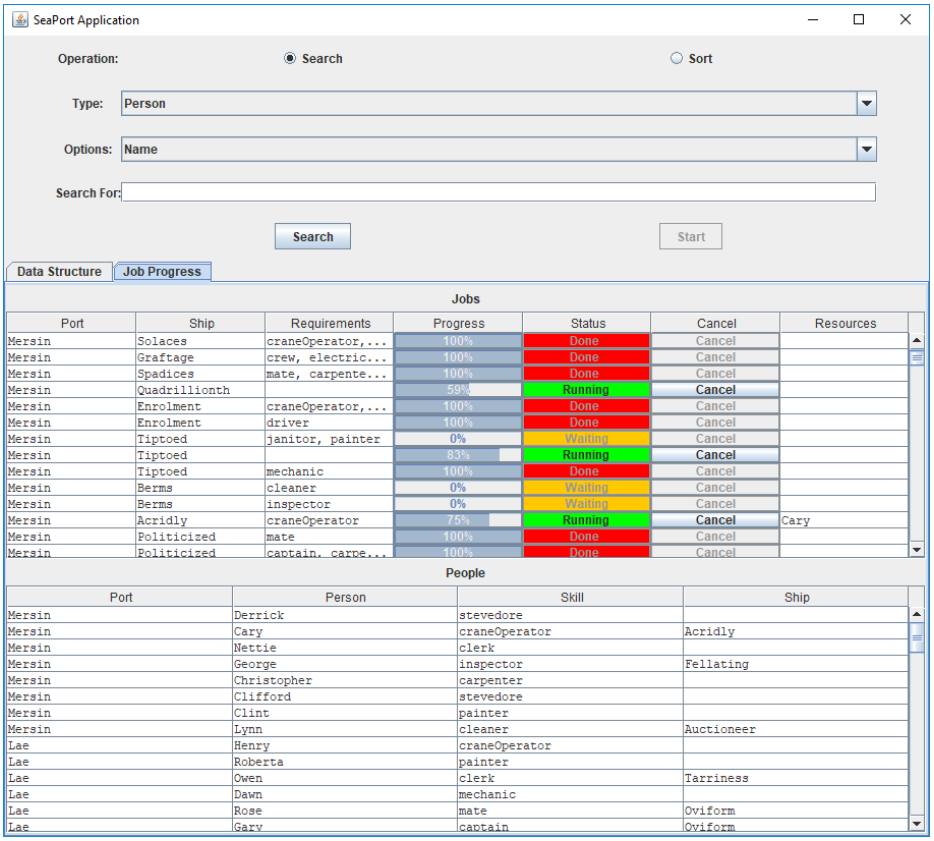
- Click on 'Cancel' button for a few jobs:
  - Job running at port Mersin for ship Quadrillionth
  - Second job running at port Mersin for ship Tiptoed
  - Job running at port Mersin for ship Acridly (using resource Cary)

This test case tests the functionality of the 'Cancel' button. When the 'Cancel' button is pushed for a job, the job is finished (because it is no longer needed). That job also releases the resources (people), that it used, if any, to progress. Once all jobs are done for that ship (including the canceled one(s)), the ship should still be removed from the dock at which it was located.

The expected results for this case are as follows:

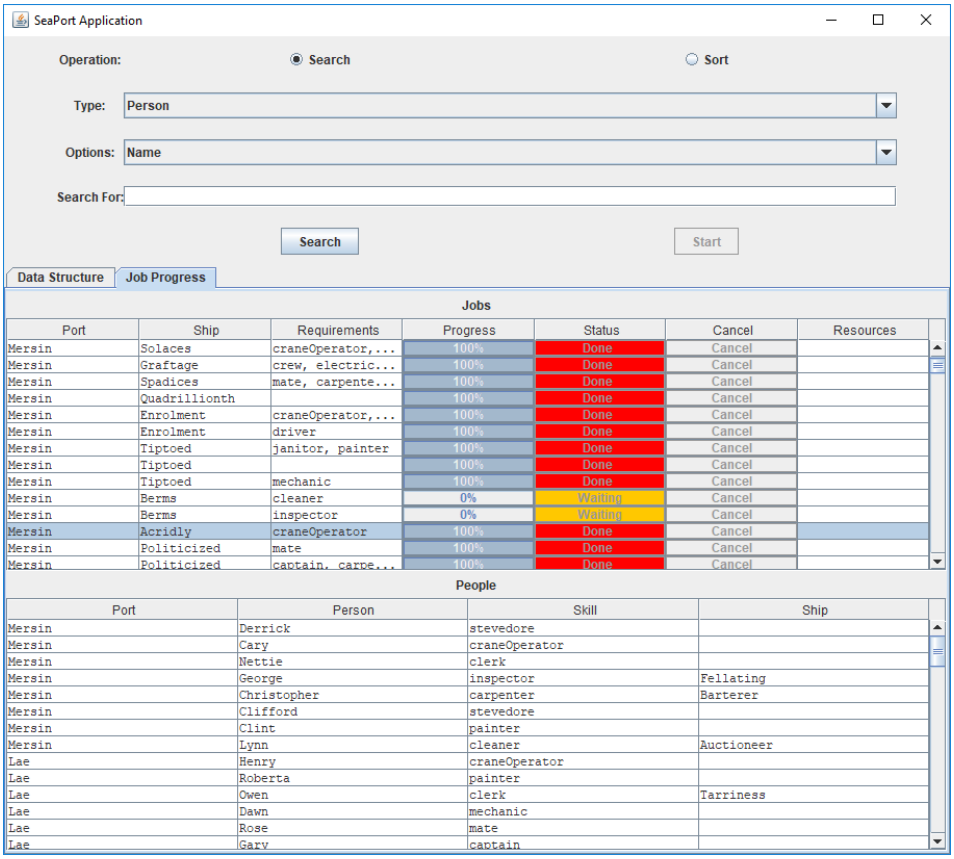
All canceled jobs will display a status of 'Done', the resources (people) will be removed from the table, and they will be removed from the docks in which they were located in the JTree.


SeaPort Application
Operation:
Search
Sort
Type:
Person
Options:
Name
Search For:
Search
Start
Data Structure
Job Progress
Open
Look In:
SeaPortProgram4
build
dist
nbproject
src
test
aSPad.txt
aSPad2.txt
aSPad3.txt
aSPad4.txt
aSPad5.txt
build.xml
manifest.mf
File Name:
aSPad2.txt
Files of Type:
All Files
Open
Cancel


SeaPort Application
Operation:
Search
Sort
Type:
Person
Options:
Name
Search For:
Search
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Mersin | Solaces | craneOperator,... | 100% | Done | Cancel |
Mersin | Graftage | crew, electric... | 100% | Done | Cancel |
Mersin | Spadices | mate, carpente... | 100% | Done | Cancel |
Mersin | Quadrillionth | | 59% | Running | Cancel |
Mersin | Enrolment | craneOperator,... | 100% | Done | Cancel |
Mersin | Enrolment | driver | 100% | Done | Cancel |
Mersin | Tiptoed | janitor, painter | 0% | Waiting | Cancel |
Mersin | Tiptoed | | 83% | Running | Cancel |
Mersin | Tiptoed | mechanic | 100% | Done | Cancel |
Mersin | Berms | cleaner | 0% | Waiting | Cancel |
Mersin | Berms | inspector | 0% | Waiting | Cancel |
Mersin | Acridly | craneOperator | 75% | Running | Cancel | Cary
Mersin | Politicized | mate | 100% | Done | Cancel |
Mersin | Politicized | captain, carpe... | 100% | Done | Cancel |
People
Port | Person | Skill | Ship
Mersin | Derrick | stevedore |
Mersin | Cary | craneOperator | Acridly
Mersin | Nettie | clerk |
Mersin | George | inspector | Fellating
Mersin | Christopher | carpenter |
Mersin | Clifford | stevedore |
Mersin | Clint | painter |
Mersin | Lynn | cleaner | Auctioneer
Lae | Henry | craneOperator |
Lae | Roberta | painter |
Lae | Owen | clerk | Tarriness
Lae | Dawn | mechanic |
Lae | Rose | mate | Oviform
Lae | Gary | captain | Oviform

<u>Test Case #7—Output:</u> Actual results match the expected output in our test table.

Note: Cary was removed from the job running for ship Acridly.



Test Case #8—Input:

File chosen: aSPad2.txt

Operation: Search

Type: Dock

Options: Name

Search For: PIER_15

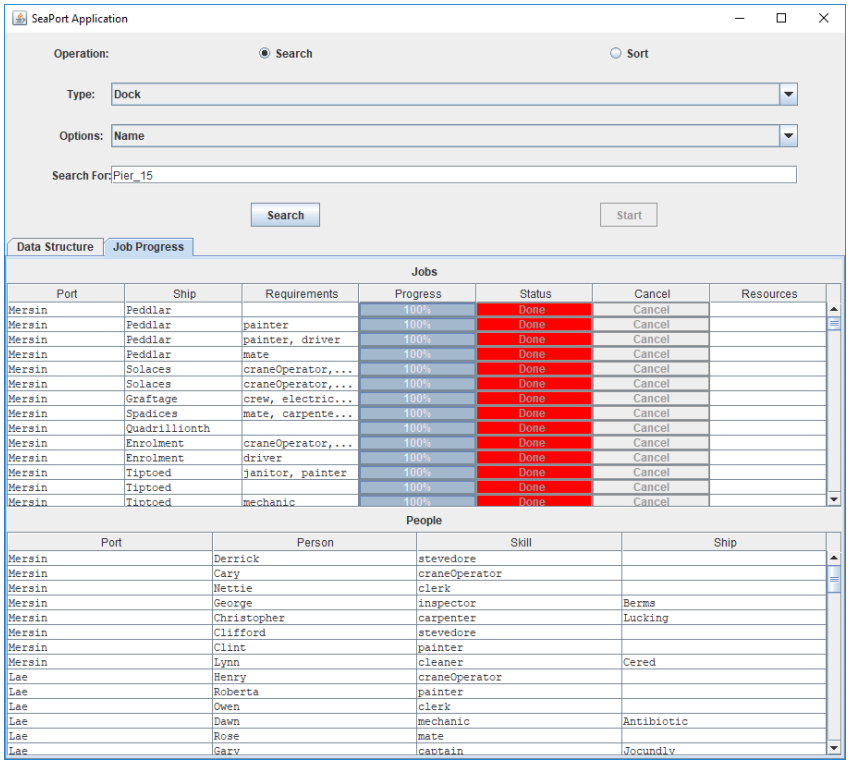
Button Selected: Search

This test case tests the Dock and Name search options of the GUI with upper case letters. Any dock searched should be empty since jobs have finished running. This case ensures that this functionality remains from Project 3.

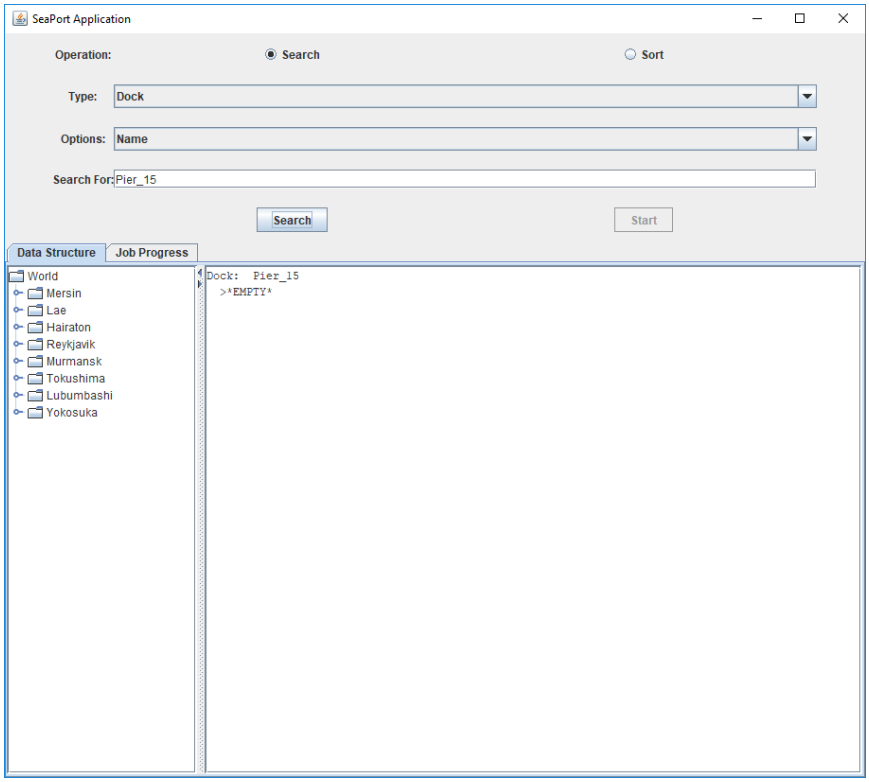
The expected results for this case are as follows:

Dock: Pier_15

>*EMPTY*



<u>Test Case #8—Output:</u> Actual results match the expected output in our test table.



Test Case #9—Input:

File chosen: aSPad2.txt

Operation: N/A

Type: N/A

Options: N/A

Search For: N/A

Button Selected: N/A

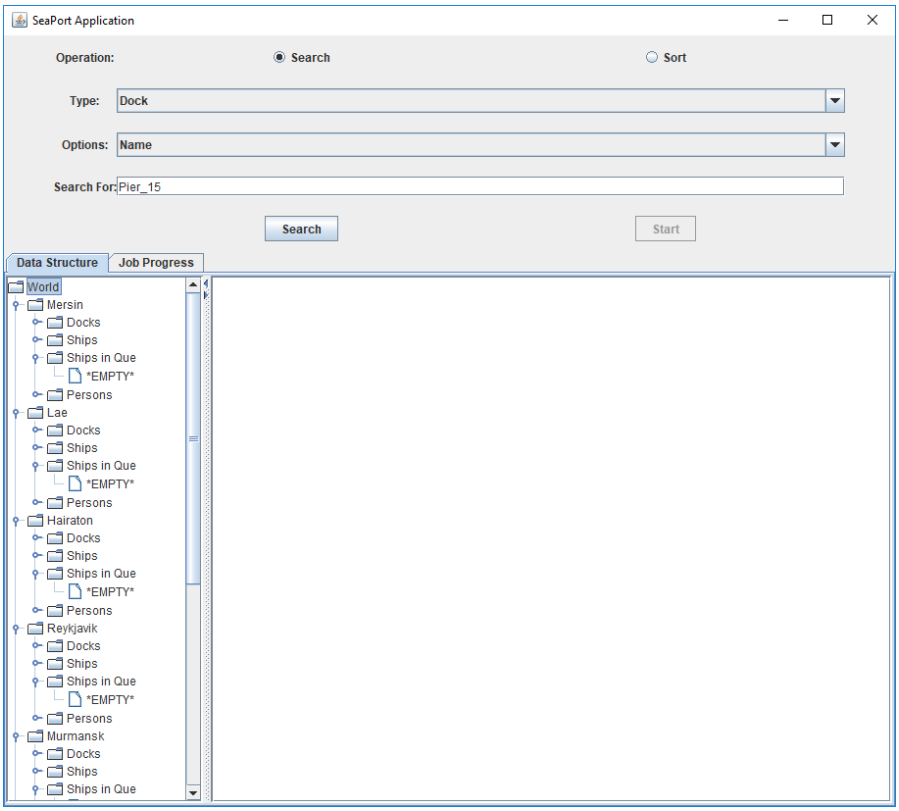
*Examining Ships in Que node of JTree on left side of JSplitPane under 'Data Structure' tab

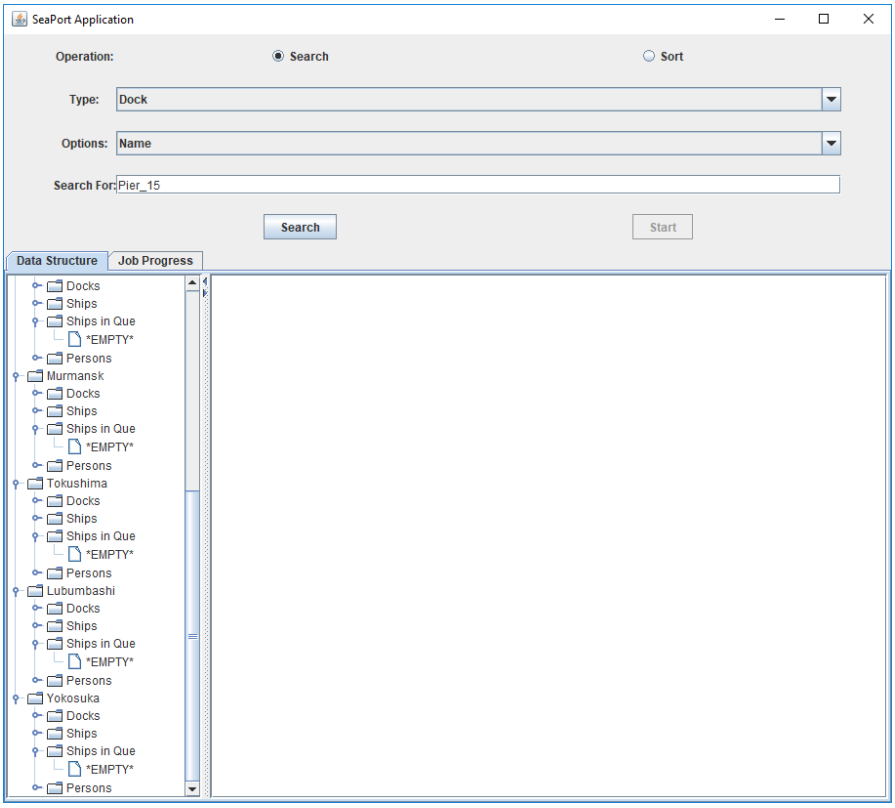
This test case tests the removal of ships from the Ships in Que. All ships listed under this node of the JTree should be removed once the jobs have finished running. This case ensures that this functionality remains from Project 3.

The expected results for this case are as follows:

All of the Ships in Que nodes of the JTree will be marked as *EMPTY*.

Test Case #9—Output: Actual results match the expected output in our test table.





Test Case #10—Input:

File chosen: *aSPad3.txt*

Operation: N/A

Type: N/A

Options: N/A

Search For: N/A

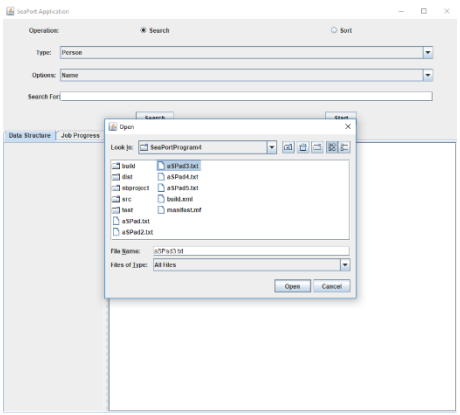
Button Selected: Start

- Click on 'Status' button when job is running (says "Running") for a few jobs:
  - Job at port Linyi for ship Benched
  - Job at port Linyi for ship Imperturbability (utilizing resources Johnnie and Mona)
  - Job at port Linyi for ship Jackfishes (utilizing resources Wayne and Sue)

This test case tests the functionality of the 'Status' button. When the 'Status' button is pushed for a running job, the job is paused/suspended. Ships of suspended jobs should remain at the docks at which they are located. Resources (people) being utilized to run the job are not released. Any other jobs that have not been run for that ship remain in waiting.

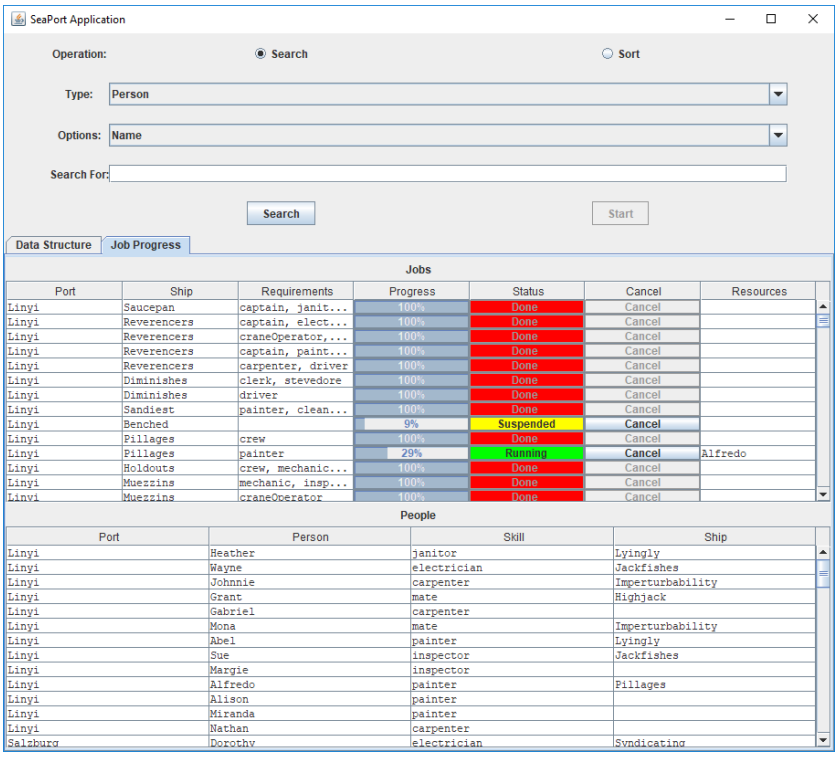
The expected results for this case are as follows:

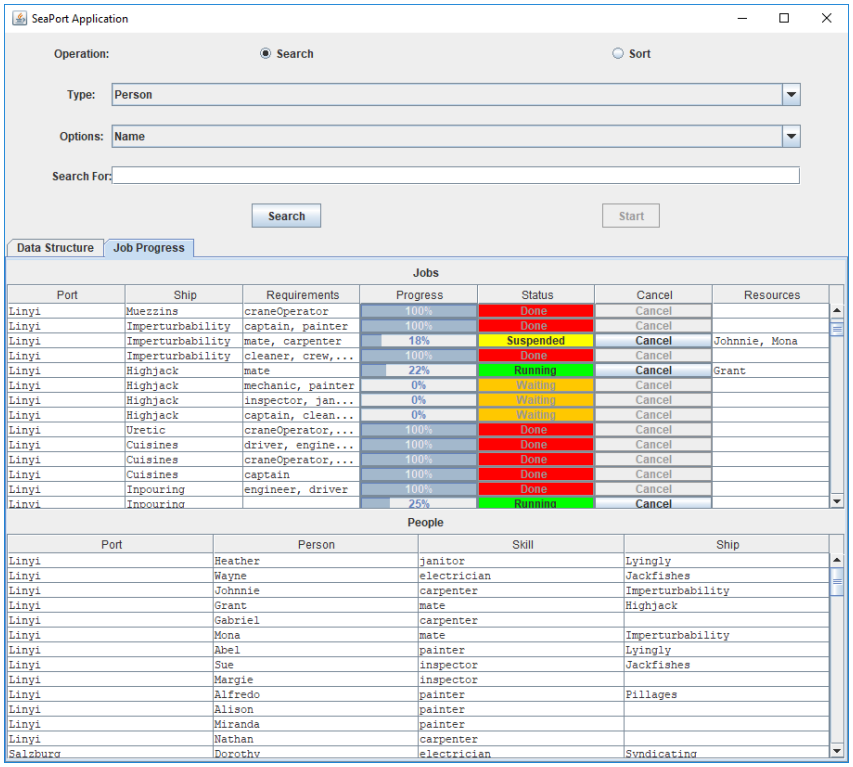
All running jobs whose 'Status' button is clicked will be paused and the 'Status' button will turn yellow and be marked "Suspended." They will not release their resources (people). Also, they will remain in the dock, as depicted in the JTree.

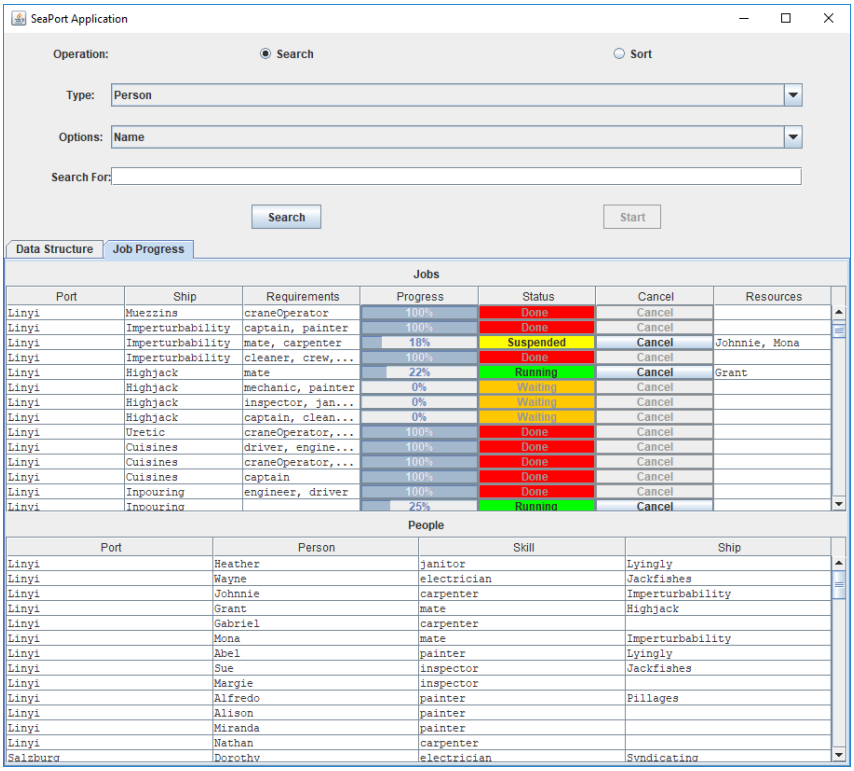


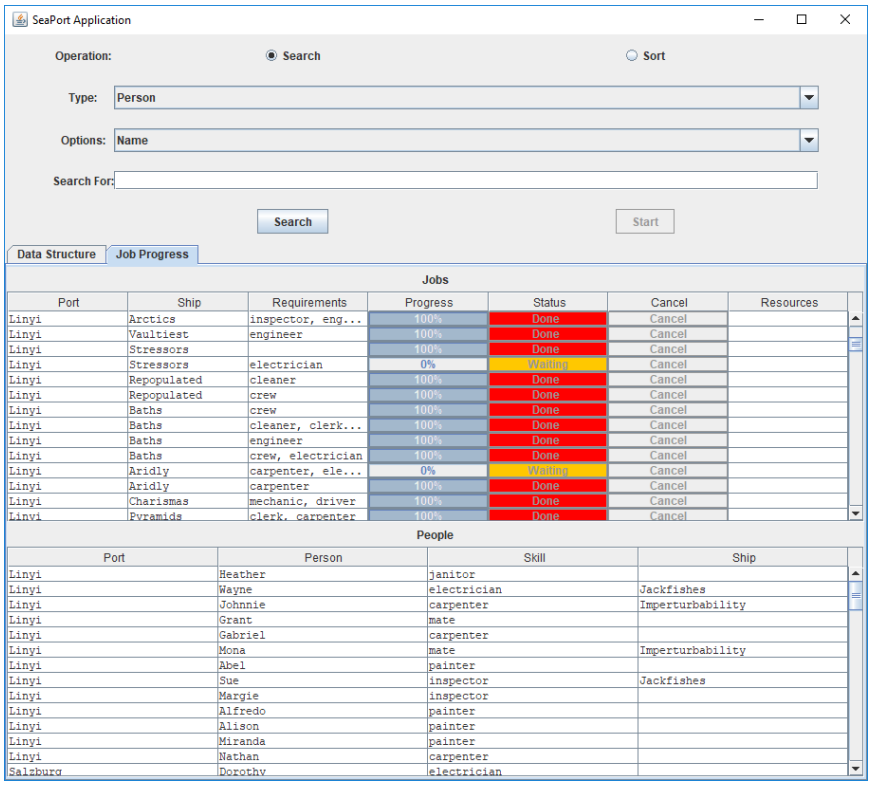
<u>Test Case #10—Output:</u> Actual results match the expected output in our test table.

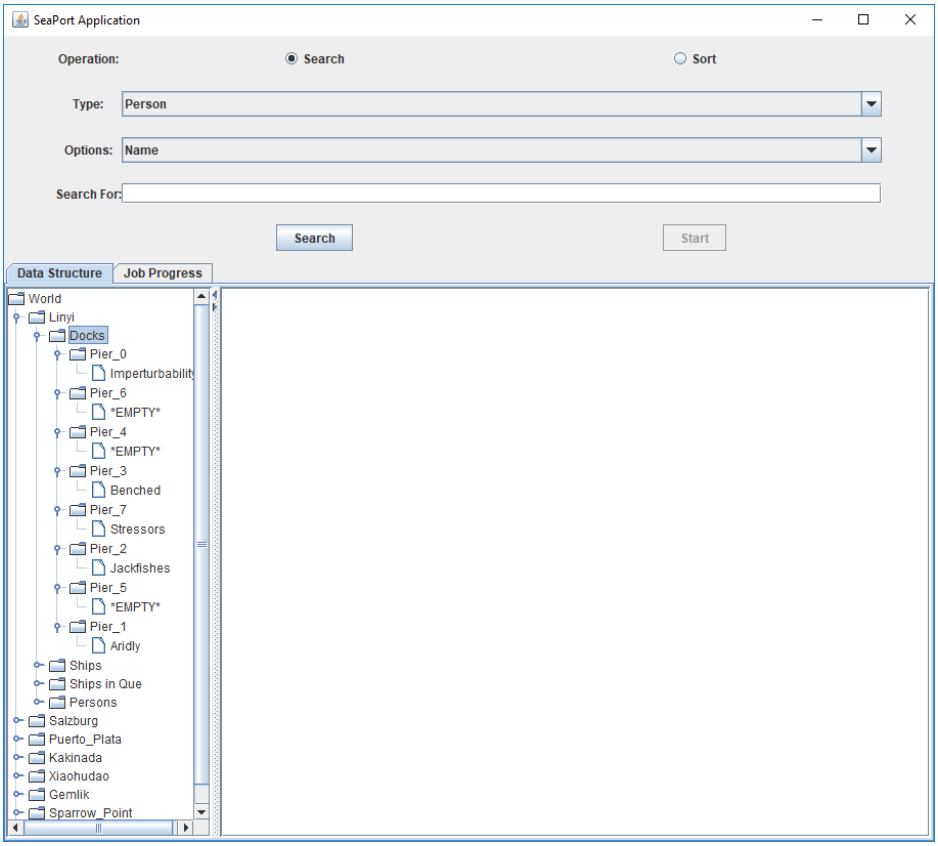
Note: Two other jobs (i.e. for ships Stressors and Aridly) are left in waiting because they need resources from the suspended jobs at port Linyi. They remain located at their respective docks.






SeaPort Application
Operation:
Search
Sort
Type:
Person
Options:
Name
Search For:
Search
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Linyi | Muezzins | craneOperator | 100% | Done | Cancel |
Linyi | Imperturbability | captain, painter | 100% | Done | Cancel |
Linyi | Imperturbability | mate, carpenter | 18% | Suspended | Cancel | Johnnie, Mona
Linyi | Imperturbability | cleaner, crew,... | 100% | Done | Cancel |
Linyi | Highjack | mate | 22% | Running | Cancel | Grant
Linyi | Highjack | mechanic, painter | 0% | Waiting | Cancel |
Linyi | Highjack | inspector, jan... | 0% | Waiting | Cancel |
Linyi | Highjack | captain, clean... | 0% | Waiting | Cancel |
Linyi | Uretic | craneOperator,... | 100% | Done | Cancel |
Linyi | Cuisines | driver, engine... | 100% | Done | Cancel |
Linyi | Cuisines | craneOperator,... | 100% | Done | Cancel |
Linyi | Cuisines | captain | 100% | Done | Cancel |
Linyi | Inpouring | engineer, driver | 100% | Done | Cancel |
Linyi | Inpouring | | 25% | Running | Cancel |
People
Port | Person | Skill | Ship
Linyi | Heather | janitor | Lyingly
Linyi | Wayne | electrician | Jackfishes
Linyi | Johnnie | carpenter | Imperturbability
Linyi | Grant | mate | Highjack
Linyi | Gabriel | carpenter |
Linyi | Mona | mate | Imperturbability
Linyi | Abel | painter | Lyingly
Linyi | Sue | inspector | Jackfishes
Linyi | Margie | inspector |
Linyi | Alfredo | painter | Pillages
Linyi | Alison | painter |
Linyi | Miranda | painter |
Linyi | Nathan | carpenter |
Salzburg | Dorothy | electrician | Syndicating


SeaPort Application
Operation:
Search
Sort
Type:
Person
Options:
Name
Search For:
Search
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Linyi | Arctics | inspector, eng... | 100% | Done | Cancel |
Linyi | Vaultiest | engineer | 100% | Done | Cancel |
Linyi | Stressors | | 100% | Done | Cancel |
Linyi | Stressors | electrician | 0% | Waiting | Cancel |
Linyi | Repopulated | cleaner | 100% | Done | Cancel |
Linyi | Repopulated | crew | 100% | Done | Cancel |
Linyi | Baths | crew | 100% | Done | Cancel |
Linyi | Baths | cleaner, clerk... | 100% | Done | Cancel |
Linyi | Baths | engineer | 100% | Done | Cancel |
Linyi | Baths | crew, electrician | 100% | Done | Cancel |
Linyi | Aridly | carpenter, ele... | 0% | Waiting | Cancel |
Linyi | Aridly | carpenter | 100% | Done | Cancel |
Linyi | Charismas | mechanic, driver | 100% | Done | Cancel |
Linyi | Pyramids | clerk, carpenter | 100% | Done | Cancel |
People
Port | Person | Skill | Ship
Linyi | Heather | janitor |
Linyi | Wayne | electrician | Jackfishes
Linyi | Johnnie | carpenter | Imperturbability
Linyi | Grant | mate |
Linyi | Gabriel | carpenter |
Linyi | Mona | mate | Imperturbability
Linyi | Abel | painter |
Linyi | Sue | inspector | Jackfishes
Linyi | Margie | inspector |
Linyi | Alfredo | painter |
Linyi | Alison | painter |
Linyi | Miranda | painter |
Linyi | Nathan | carpenter |
Salzburg | Dorothy | electrician |


SeaPort Application
Operation:
Search
Sort
Type:
Person
Options:
Name
Search For:
Search
Start
Data Structure
Job Progress
World
Linyi
Docks
Pier_0
Imperturbability
Pier_6
EMPTY
Pier_4
EMPTY
Pier_3
Benched
Pier_7
Stressors
Pier_2
Jackfishes
Pier_5
EMPTY
Pier_1
Aridly
Ships
Ships in Que
Persons
Salzburg
Puerto_Plata
Kakinada
Xiaohudao
Gemlik
Sparrow_Point

Test Case #11—Input:

File chosen: *aSPad4.txt*

Operation: N/A

Type: N/A

Options: N/A

Search For: N/A

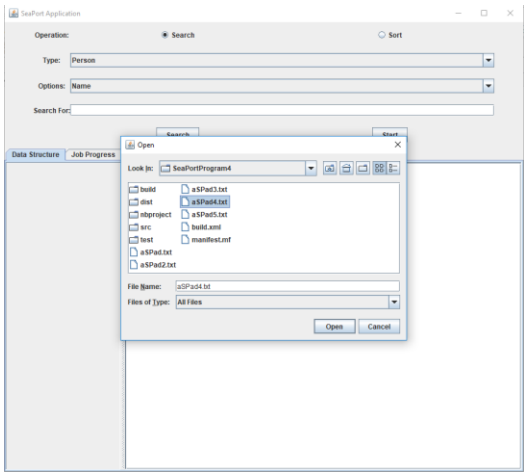
Button Selected: Start

*Examining the resources (people) in the 'Jobs' table to see that they are moving when jobs are completed and ensuring that their ship name changes in the 'People' table.

This test case tests the functionality of the resources being utilized in the multithreading process to ensure that these resources (people) are being moved properly and that this is being displayed correctly in the tables in the 'Job Progress' tab.

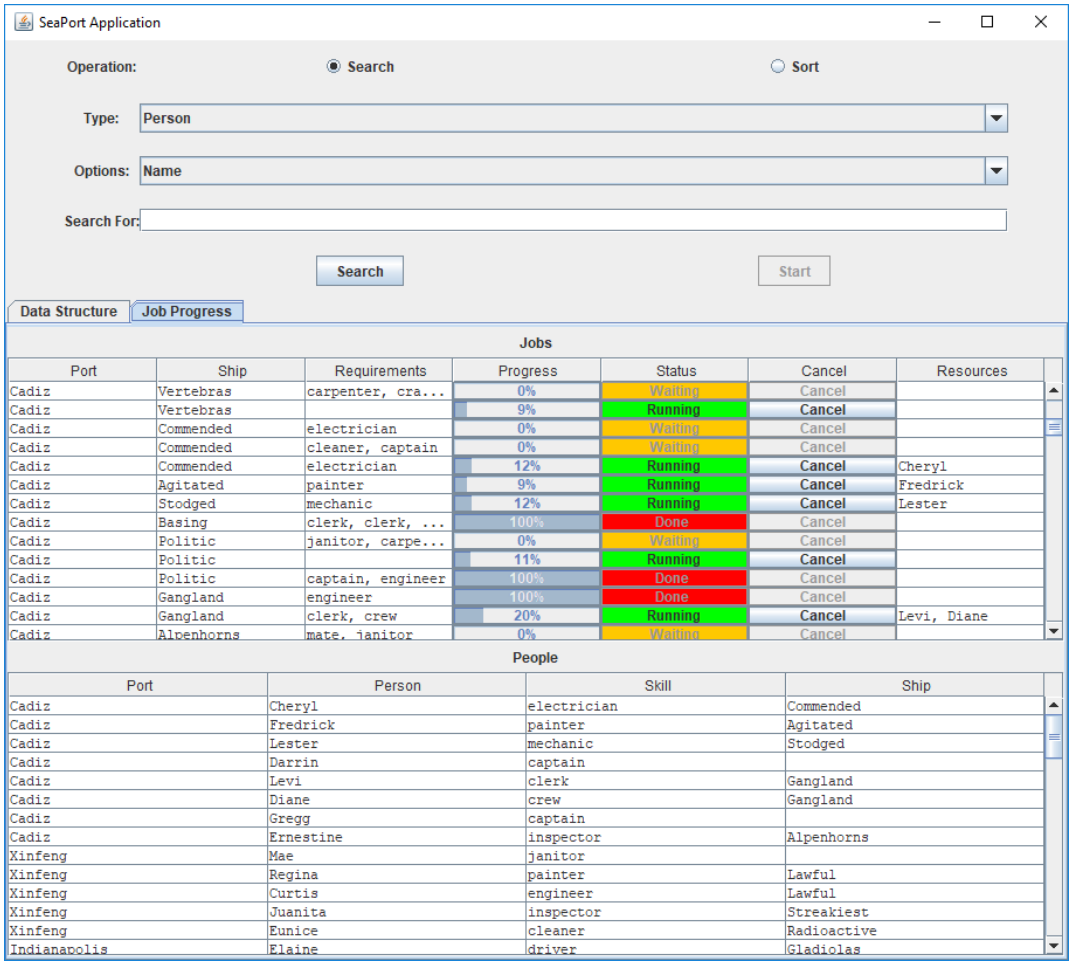
The expected results for this case are as follows:

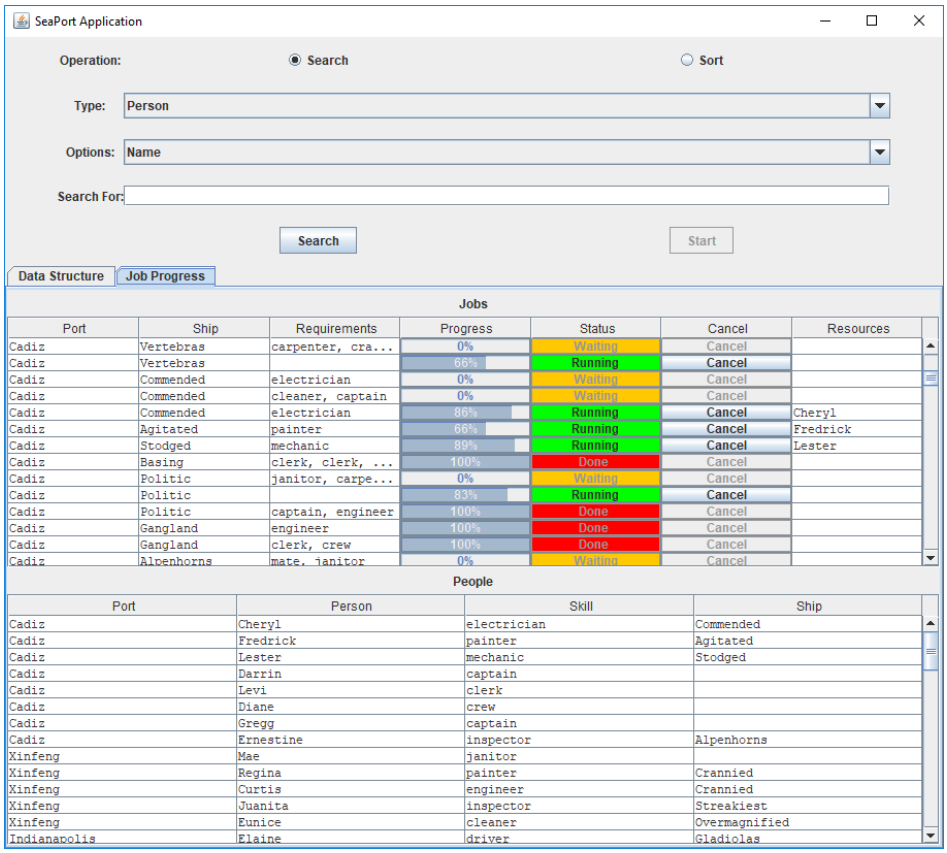
In the 'Jobs' table, resources (people) being used by running jobs will be moved once those jobs are complete to other jobs that were previously waiting to progress. Also, in the 'Jobs' table, the ship name where the person is working will change when the person switches jobs.

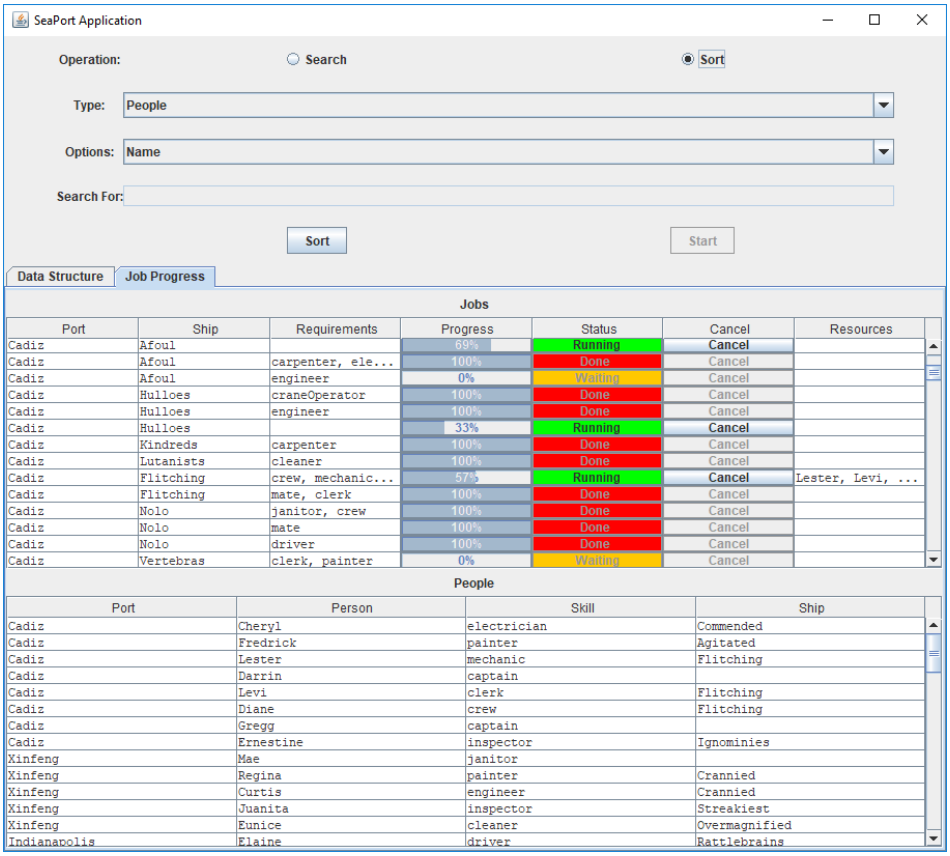


<u>Test Case #11—Output:</u> Actual results match the expected output in our test table.

Note: These screenshots focus on the following resources (people): Cheryl, Frederick, Lester, Levi, and Diane. They move from ship to ship to work on jobs and this is displayed correctly in both the 'Jobs' and 'People' tables.




SeaPort Application
Operation:
Search
Sort
Type:
Person
Options:
Name
Search For:
Search
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Cadiz | Vertebras | carpenter, cra... | 0% | Waiting | Cancel |
Cadiz | Vertebras | | 66% | Running | Cancel |
Cadiz | Commended | electrician | 0% | Waiting | Cancel |
Cadiz | Commended | cleaner, captain | 0% | Waiting | Cancel |
Cadiz | Commended | electrician | 86% | Running | Cancel | Cheryl
Cadiz | Agitated | painter | 66% | Running | Cancel | Fredrick
Cadiz | Stodged | mechanic | 89% | Running | Cancel | Lester
Cadiz | Basing | clerk, clerk, ... | 100% | Done | Cancel |
Cadiz | Politic | janitor, carpe... | 0% | Waiting | Cancel |
Cadiz | Politic | | 83% | Running | Cancel |
Cadiz | Politic | captain, engineer | 100% | Done | Cancel |
Cadiz | Gangland | engineer | 100% | Done | Cancel |
Cadiz | Gangland | clerk, crew | 100% | Done | Cancel |
Cadiz | Alpenhorns | mate, janitor | 0% | Waiting | Cancel |
People
Port | Person | Skill | Ship
Cadiz | Cheryl | electrician | Commended
Cadiz | Fredrick | painter | Agitated
Cadiz | Lester | mechanic | Stodged
Cadiz | Darrin | captain |
Cadiz | Levi | clerk |
Cadiz | Diane | crew |
Cadiz | Gregg | captain |
Cadiz | Ernestine | inspector | Alpenhorns
Xinfeng | Mae | janitor |
Xinfeng | Regina | painter | Crannied
Xinfeng | Curtis | engineer | Crannied
Xinfeng | Juanita | inspector | Streakiest
Xinfeng | Eunice | cleaner | Overmagnified
Indianapolis | Elaine | driver | Gladiolas


SeaPort Application
Operation:
Search
Sort
Type:
People
Options:
Name
Search For:
Sort
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Cadiz | Afoul | | 69% | Running | Cancel |
Cadiz | Afoul | carpenter, ele... | 100% | Done | Cancel |
Cadiz | Afoul | engineer | 0% | Waiting | Cancel |
Cadiz | Hulloes | craneOperator | 100% | Done | Cancel |
Cadiz | Hulloes | engineer | 100% | Done | Cancel |
Cadiz | Hulloes | | 33% | Running | Cancel |
Cadiz | Kindreds | carpenter | 100% | Done | Cancel |
Cadiz | Lutanists | cleaner | 100% | Done | Cancel |
Cadiz | Flitching | crew, mechanic... | 57% | Running | Cancel | Lester, Levi, ...
Cadiz | Flitching | mate, clerk | 100% | Done | Cancel |
Cadiz | Nolo | janitor, crew | 100% | Done | Cancel |
Cadiz | Nolo | mate | 100% | Done | Cancel |
Cadiz | Nolo | driver | 100% | Done | Cancel |
Cadiz | Vertebras | clerk, painter | 0% | Waiting | Cancel |
People
Port | Person | Skill | Ship
Cadiz | Cheryl | electrician | Commended
Cadiz | Fredrick | painter | Agitated
Cadiz | Lester | mechanic | Flitching
Cadiz | Darrin | captain |
Cadiz | Levi | clerk | Flitching
Cadiz | Diane | crew | Flitching
Cadiz | Gregg | captain |
Cadiz | Ernestine | inspector | Ignominies
Xinfeng | Mae | janitor |
Xinfeng | Regina | painter | Crannied
Xinfeng | Curtis | engineer | Crannied
Xinfeng | Juanita | inspector | Streakiest
Xinfeng | Eunice | cleaner | Overmagnified
Indianapolis | Elaine | driver | Rattlebrains

SeaPort Application

Operation: Search Sort

Type: People

Options: Name

Search For:

Sort Start

Data Structure | Job Progress

Jobs

Port	Ship	Requirements	Progress	Status	Cancel	Resources
Cadiz	Flitching	crew, mechanic...	100%	Done	Cancel	
Cadiz	Flitching	mate, clerk	100%	Done	Cancel	
Cadiz	Nolo	janitor, crew	100%	Done	Cancel	
Cadiz	Nolo	mate	100%	Done	Cancel	
Cadiz	Nolo	driver	100%	Done	Cancel	
Cadiz	Vertebras	clerk, painter	47%	Running	Cancel	Fredrick, Levi
Cadiz	Vertebras	carpenter, cra...	100%	Done	Cancel	
Cadiz	Vertebras		100%	Done	Cancel	
Cadiz	Commended	electrician	71%	Running	Cancel	Cheryl
Cadiz	Commended	cleaner, captain	0%	Waiting	Cancel	
Cadiz	Commended	electrician	100%	Done	Cancel	
Cadiz	Agitated	painter	100%	Done	Cancel	
Cadiz	Stodged	mechanic	100%	Done	Cancel	
Cadiz	Basing	clerk, clerk, ...	100%	Done	Cancel	

People

Port	Person	Skill	Ship
Cadiz	Cheryl	electrician	Commended
Cadiz	Fredrick	painter	Vertebras
Cadiz	Lester	mechanic	Distances
Cadiz	Darrin	captain	
Cadiz	Levi	clerk	Vertebras
Cadiz	Diane	crew	
Cadiz	Gregg	captain	
Cadiz	Ernestine	inspector	
Xinfeng	Mae	janitor	
Xinfeng	Regina	painter	Crannied
Xinfeng	Curtis	engineer	Crannied
Xinfeng	Juanita	inspector	
Xinfeng	Eunice	cleaner	
Indianapolis	Elaine	driver	Rattlebrains

SeaPort Application

Operation: Search Sort

Type: People

Options: Name

Search For:

Sort Start

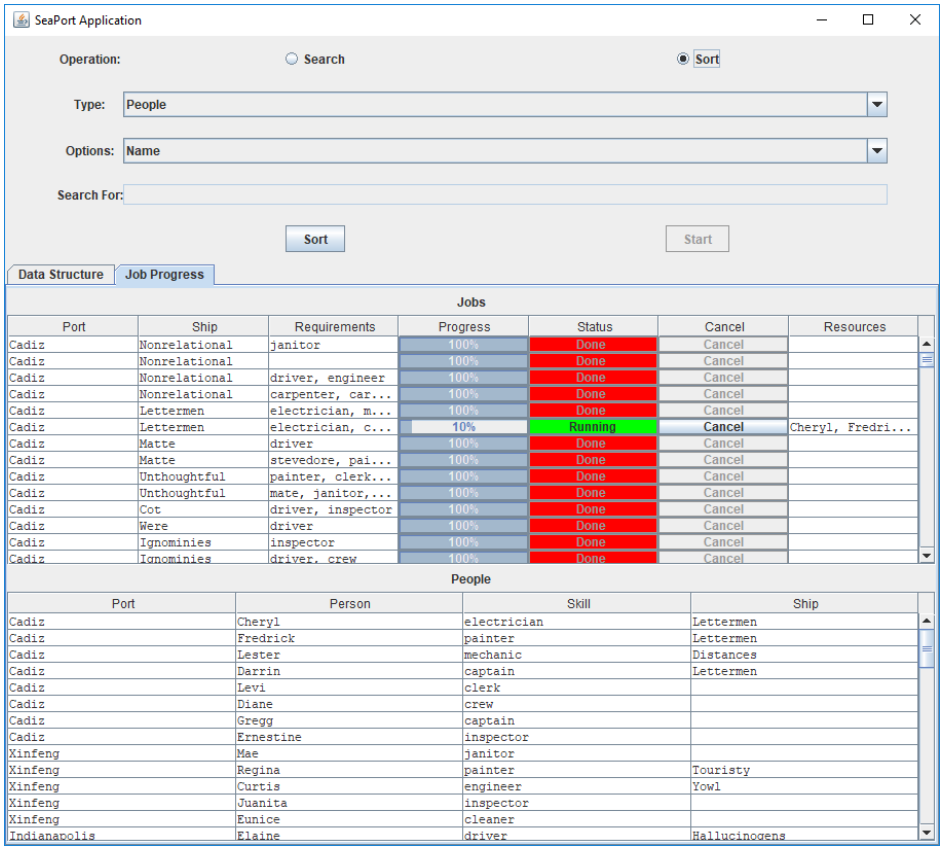
Data Structure | Job Progress

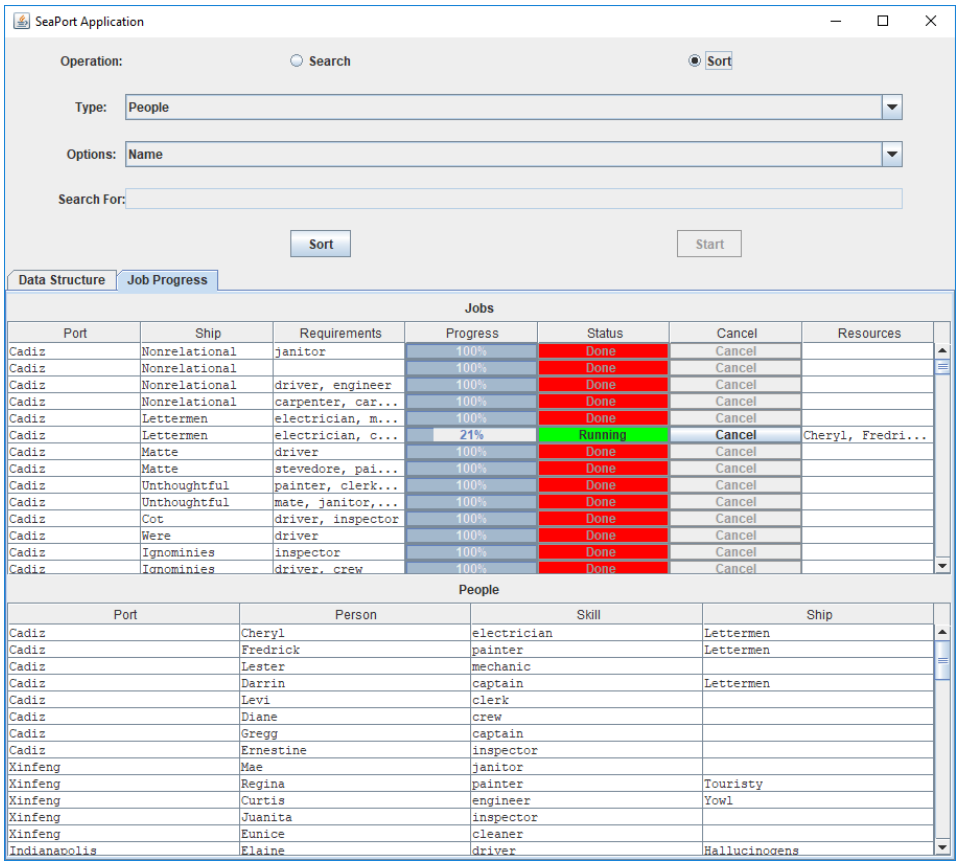
Jobs

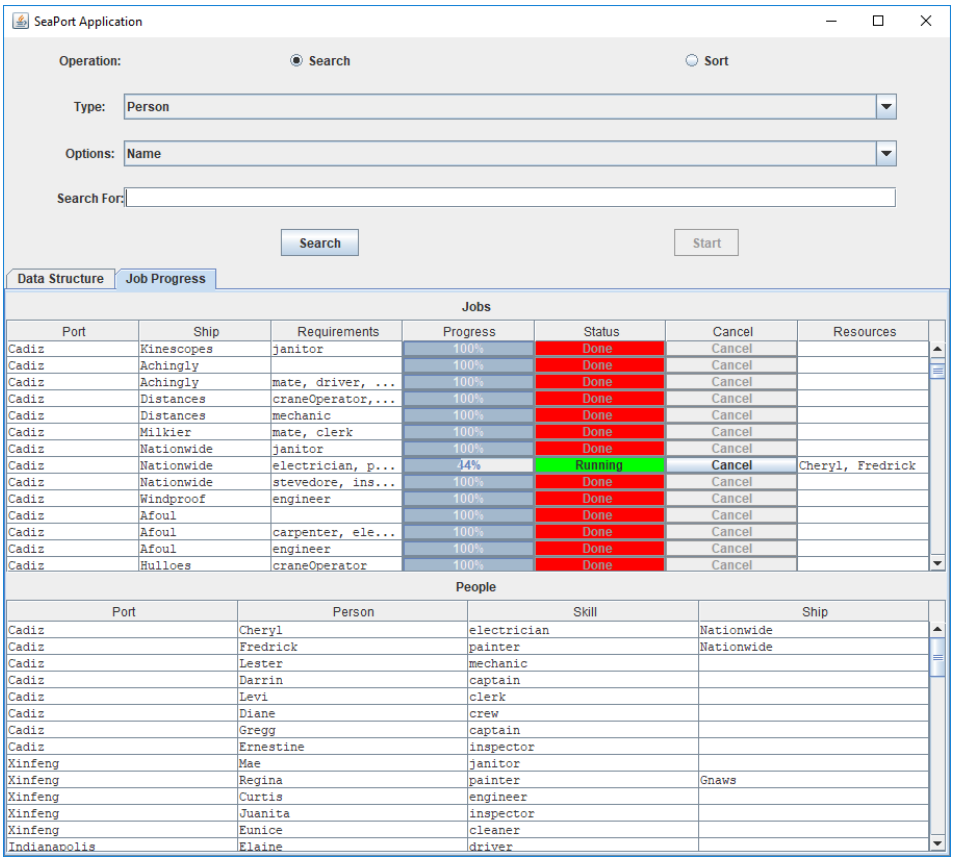
Port	Ship	Requirements	Progress	Status	Cancel	Resources
Cadiz	Ignominies	inspector	100%	Done	Cancel	
Cadiz	Ignominies	driver, crew	100%	Done	Cancel	
Cadiz	Kinescopes		100%	Done	Cancel	
Cadiz	Kinescopes	janitor	100%	Done	Cancel	
Cadiz	Achingly		100%	Done	Cancel	
Cadiz	Achingly	mate, driver, ...	100%	Done	Cancel	
Cadiz	Distances	craneOperator,...	100%	Done	Cancel	
Cadiz	Distances	mechanic	64%	Running	Cancel	Lester
Cadiz	Milkier	mate, clerk	100%	Done	Cancel	
Cadiz	Nationwide	janitor	100%	Done	Cancel	
Cadiz	Nationwide	electrician, p...	0%	Waiting	Cancel	
Cadiz	Nationwide	stevedore, ins...	100%	Done	Cancel	
Cadiz	Windproof	engineer	100%	Done	Cancel	
Cadiz	Afoul		100%	Done	Cancel	

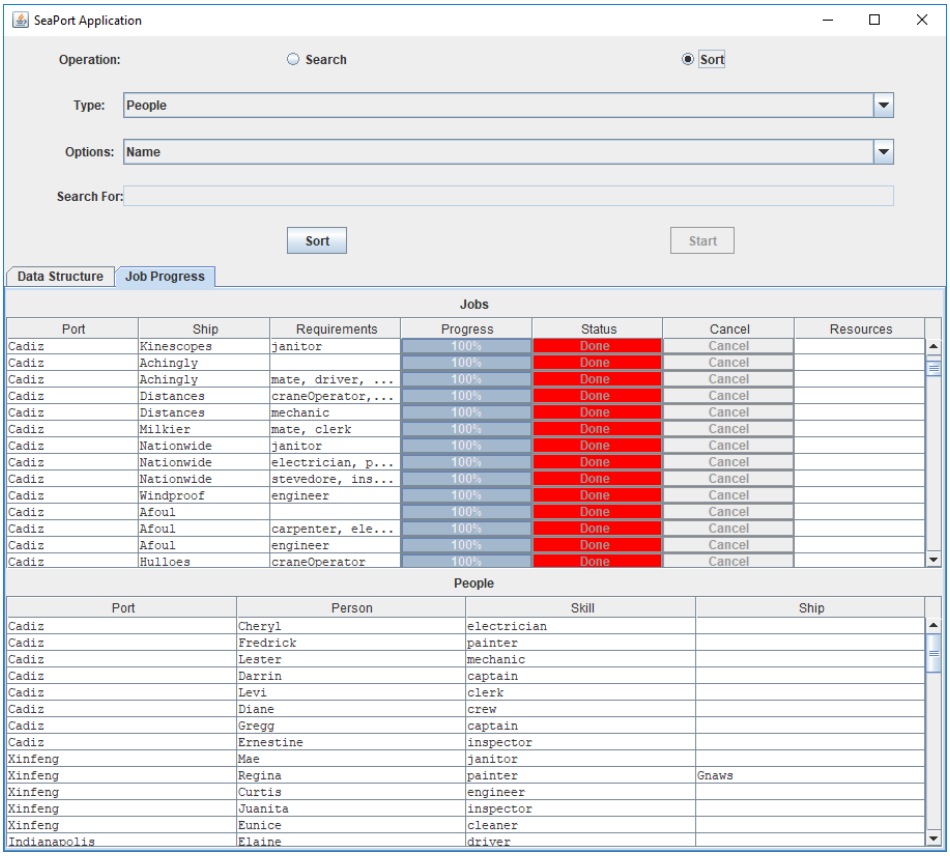
People

Port	Person	Skill	Ship
Cadiz	Cheryl	electrician	Commended
Cadiz	Fredrick	painter	Vertebras
Cadiz	Lester	mechanic	Distances
Cadiz	Darrin	captain	
Cadiz	Levi	clerk	Vertebras
Cadiz	Diane	crew	
Cadiz	Gregg	captain	
Cadiz	Ernestine	inspector	
Xinfeng	Mae	janitor	
Xinfeng	Regina	painter	Crannied
Xinfeng	Curtis	engineer	Crannied
Xinfeng	Juanita	inspector	
Xinfeng	Eunice	cleaner	
Indianapolis	Elaine	driver	Rattlebrains


SeaPort Application
Operation:
Search
Sort
Type:
People
Options:
Name
Search For:
Sort
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Cadiz | Nonrelational | janitor | 100% | Done | Cancel |
Cadiz | Nonrelational | | 100% | Done | Cancel |
Cadiz | Nonrelational | driver, engineer | 100% | Done | Cancel |
Cadiz | Nonrelational | carpenter, car... | 100% | Done | Cancel |
Cadiz | Lettermen | electrician, m... | 100% | Done | Cancel |
Cadiz | Lettermen | electrician, c... | 10% | Running | Cancel | Cheryl, Fredri...
Cadiz | Matte | driver | 100% | Done | Cancel |
Cadiz | Matte | stevedore, pai... | 100% | Done | Cancel |
Cadiz | Unthoughtful | painter, clerk... | 100% | Done | Cancel |
Cadiz | Unthoughtful | mate, janitor,... | 100% | Done | Cancel |
Cadiz | Cot | driver, inspector | 100% | Done | Cancel |
Cadiz | Were | driver | 100% | Done | Cancel |
Cadiz | Ignominies | inspector | 100% | Done | Cancel |
Cadiz | Ignominies | driver, crew | 100% | Done | Cancel |
People
Port | Person | Skill | Ship
Cadiz | Cheryl | electrician | Lettermen
Cadiz | Fredrick | painter | Lettermen
Cadiz | Lester | mechanic | Distances
Cadiz | Darrin | captain | Lettermen
Cadiz | Levi | clerk |
Cadiz | Diane | crew |
Cadiz | Gregg | captain |
Cadiz | Ernestine | inspector |
Xinfeng | Mae | janitor |
Xinfeng | Regina | painter | Touristy
Xinfeng | Curtis | engineer | Yowl
Xinfeng | Juanita | inspector |
Xinfeng | Eunice | cleaner |
Indianapolis | Elaine | driver | Hallucinogens

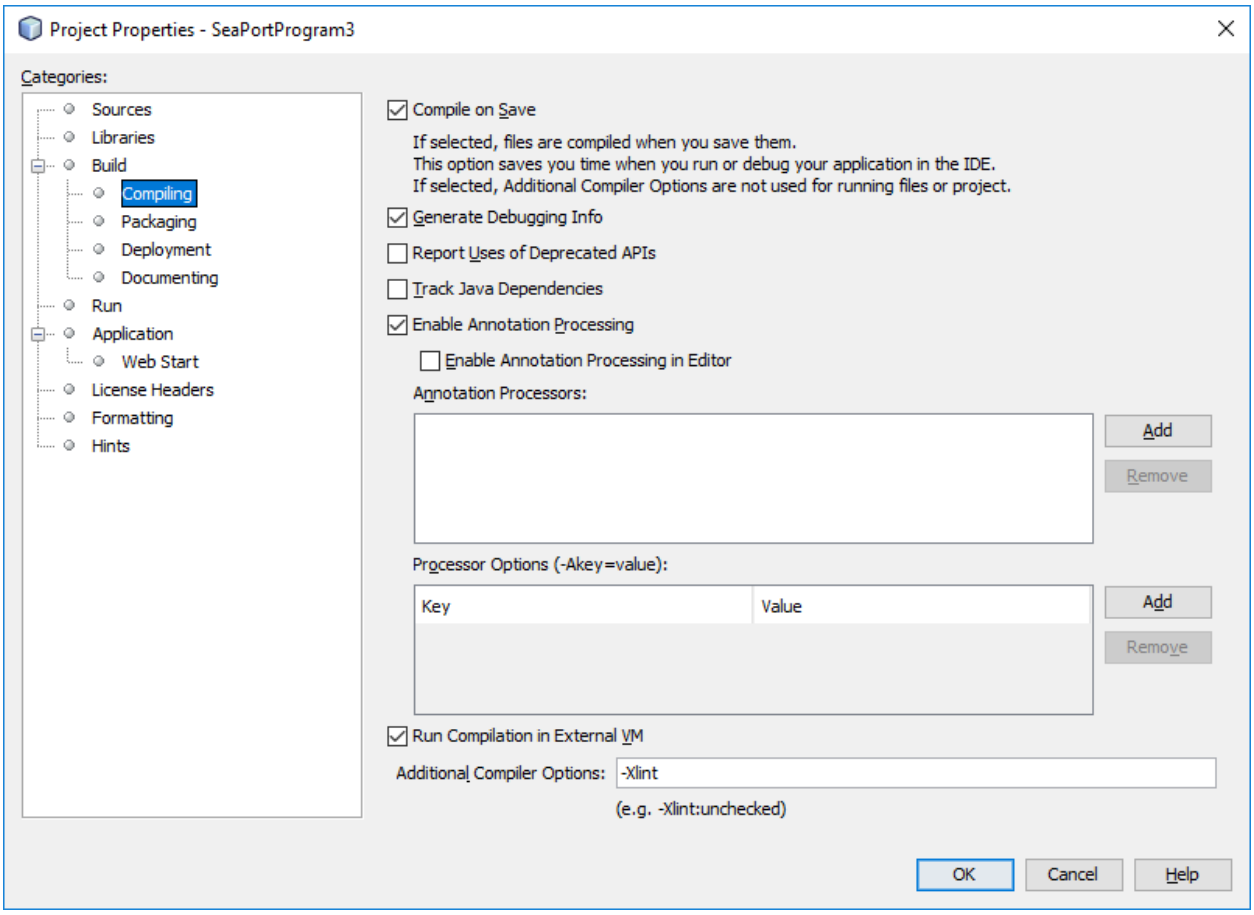

SeaPort Application
Operation:
Search
Sort
Type:
People
Options:
Name
Search For:
Sort
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Cadiz | Nonrelational | janitor | 100% | Done | Cancel |
Cadiz | Nonrelational | | 100% | Done | Cancel |
Cadiz | Nonrelational | driver, engineer | 100% | Done | Cancel |
Cadiz | Nonrelational | carpenter, car... | 100% | Done | Cancel |
Cadiz | Lettermen | electrician, m... | 100% | Done | Cancel |
Cadiz | Lettermen | electrician, c... | 21% | Running | Cancel | Cheryl, Fredri...
Cadiz | Matte | driver | 100% | Done | Cancel |
Cadiz | Matte | stevedore, pai... | 100% | Done | Cancel |
Cadiz | Unthoughtful | painter, clerk... | 100% | Done | Cancel |
Cadiz | Unthoughtful | mate, janitor,... | 100% | Done | Cancel |
Cadiz | Cot | driver, inspector | 100% | Done | Cancel |
Cadiz | Were | driver | 100% | Done | Cancel |
Cadiz | Ignominies | inspector | 100% | Done | Cancel |
Cadiz | Ignominies | driver, crew | 100% | Done | Cancel |
People
Port | Person | Skill | Ship
Cadiz | Cheryl | electrician | Lettermen
Cadiz | Fredrick | painter | Lettermen
Cadiz | Lester | mechanic |
Cadiz | Darrin | captain | Lettermen
Cadiz | Levi | clerk |
Cadiz | Diane | crew |
Cadiz | Gregg | captain |
Cadiz | Ernestine | inspector |
Xinfeng | Mae | janitor |
Xinfeng | Regina | painter | Touristy
Xinfeng | Curtis | engineer | Yowl
Xinfeng | Juanita | inspector |
Xinfeng | Eunice | cleaner |
Indianapolis | Elaine | driver | Hallucinogens

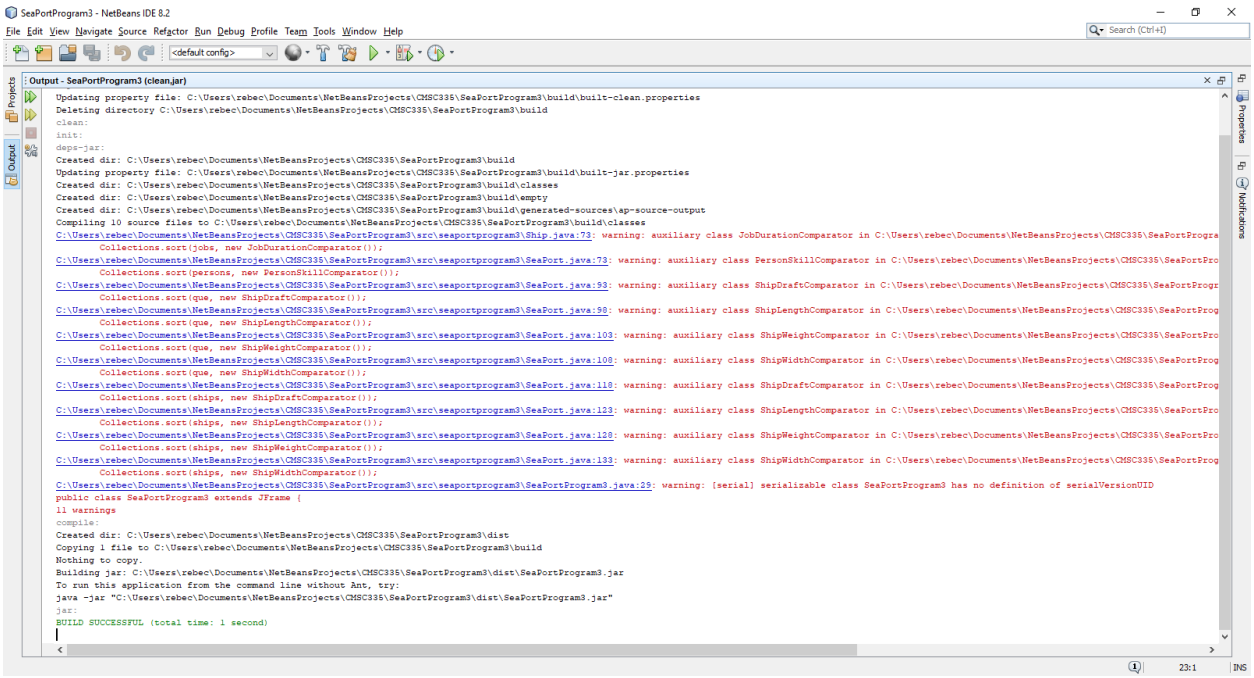

SeaPort Application
Operation:
Search
Sort
Type:
Person
Options:
Name
Search For:
Search
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Cadiz | Kinescopes | janitor | 100% | Done | Cancel |
Cadiz | Achingly | | 100% | Done | Cancel |
Cadiz | Achingly | mate, driver, ... | 100% | Done | Cancel |
Cadiz | Distances | craneOperator,... | 100% | Done | Cancel |
Cadiz | Distances | mechanic | 100% | Done | Cancel |
Cadiz | Milkier | mate, clerk | 100% | Done | Cancel |
Cadiz | Nationwide | janitor | 100% | Done | Cancel |
Cadiz | Nationwide | electrician, p... | 14% | Running | Cancel | Cheryl, Fredrick
Cadiz | Nationwide | stevedore, ins... | 100% | Done | Cancel |
Cadiz | Windproof | engineer | 100% | Done | Cancel |
Cadiz | Afoul | | 100% | Done | Cancel |
Cadiz | Afoul | carpenter, ele... | 100% | Done | Cancel |
Cadiz | Afoul | engineer | 100% | Done | Cancel |
Cadiz | Hulloes | craneOperator | 100% | Done | Cancel |
People
Port | Person | Skill | Ship
Cadiz | Cheryl | electrician | Nationwide
Cadiz | Fredrick | painter | Nationwide
Cadiz | Lester | mechanic |
Cadiz | Darrin | captain |
Cadiz | Levi | clerk |
Cadiz | Diane | crew |
Cadiz | Gregg | captain |
Cadiz | Ernestine | inspector |
Xinfeng | Mae | janitor |
Xinfeng | Regina | painter | Gnaws
Xinfeng | Curtis | engineer |
Xinfeng | Juanita | inspector |
Xinfeng | Eunice | cleaner |
Indianapolis | Elaine | driver |


SeaPort Application
Operation:
Search
Sort
Type:
People
Options:
Name
Search For:
Sort
Start
Data Structure
Job Progress
Jobs
Port | Ship | Requirements | Progress | Status | Cancel | Resources
Cadiz | Kinescopes | janitor | 100% | Done | Cancel |
Cadiz | Achingly | | 100% | Done | Cancel |
Cadiz | Achingly | mate, driver, ... | 100% | Done | Cancel |
Cadiz | Distances | craneOperator,... | 100% | Done | Cancel |
Cadiz | Distances | mechanic | 100% | Done | Cancel |
Cadiz | Milkier | mate, clerk | 100% | Done | Cancel |
Cadiz | Nationwide | janitor | 100% | Done | Cancel |
Cadiz | Nationwide | electrician, p... | 100% | Done | Cancel |
Cadiz | Nationwide | stevedore, ins... | 100% | Done | Cancel |
Cadiz | Windproof | engineer | 100% | Done | Cancel |
Cadiz | Afoul | | 100% | Done | Cancel |
Cadiz | Afoul | carpenter, ele... | 100% | Done | Cancel |
Cadiz | Afoul | engineer | 100% | Done | Cancel |
Cadiz | Hulloes | craneOperator | 100% | Done | Cancel |
People
Port | Person | Skill | Ship
Cadiz | Cheryl | electrician |
Cadiz | Fredrick | painter |
Cadiz | Lester | mechanic |
Cadiz | Darrin | captain |
Cadiz | Levi | clerk |
Cadiz | Diane | crew |
Cadiz | Gregg | captain |
Cadiz | Ernestine | inspector |
Xinfeng | Mae | janitor |
Xinfeng | Regina | painter | Gnaws
Xinfeng | Curtis | engineer |
Xinfeng | Juanita | inspector |
Xinfeng | Eunice | cleaner |
Indianapolis | Elaine | driver |

Lessons Learned:

First and foremost, I learned how to properly check *all* potential warnings for my program. For the first two projects in the SeaPort project series, I had only used -Xlint:unchecked to compile my program and resolved all of the warnings that came up. I compiled Project 3 the same way and did not see any warnings. One of the notes in my feedback for Project 3 indicated that I had 11 warnings. I did some exploring as to why I may have missed these. I did not realize that I should have actually used -Xlint to show all warnings. -Xlint:unchecked does not encompass all warning checks. When I compiled the program using just -Xlint, I noticed the same number of warnings (11) to investigate. I took care of these first before starting Project 4.



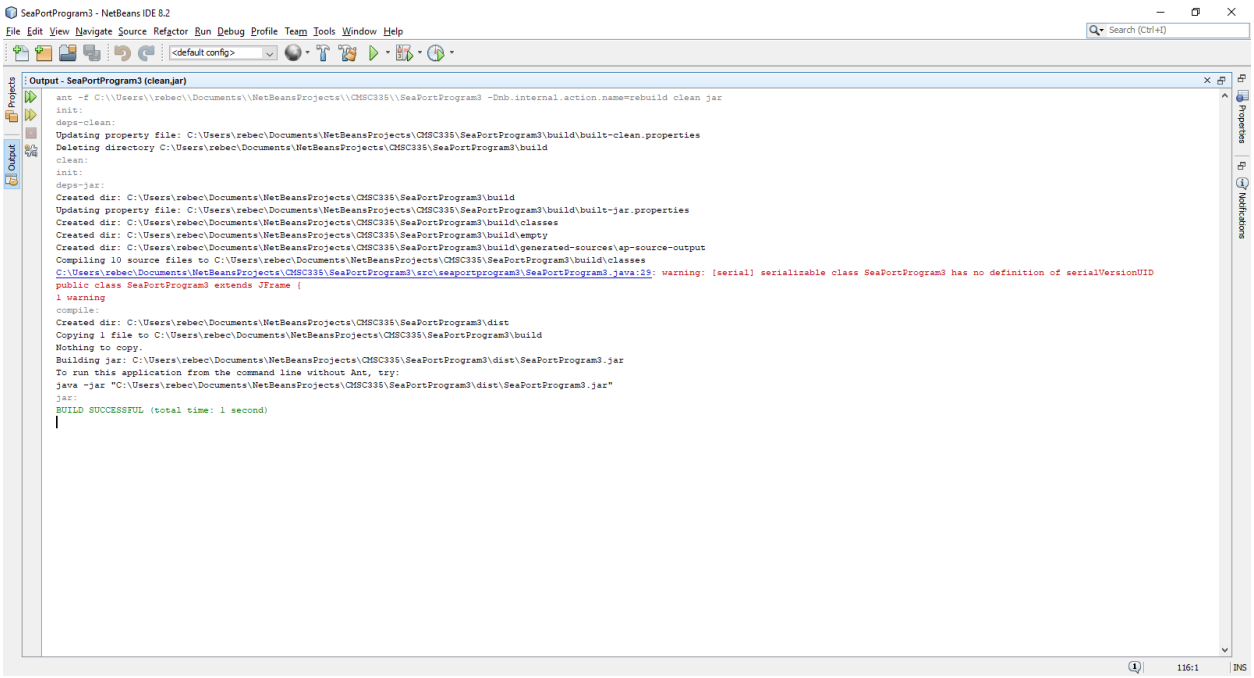


warning: auxiliary class PersonSkillComparator in C:\Users\rebec\Documents\NetBeansProjects\CMSC335\SeaPortProgram3\src\seaportprogram3\Person.java should not be accessed from outside its own source file Collections.sort(persons, new PersonSkillComparator());

Ten of the eleven warnings looked like the one above and were based on where the comparator classes were located in the program. Each comparator class is located in the file of another one of the preexisting classes. The way that I fixed the warnings was by moving the comparator classes to the file of the same class in which they were being called. After all of the comparator classes were moved, one final warning remained:

[serial] serializable class SeaPortProgram3 has no definition of serialVersionUID
public class SeaPortProgram3 extends JFrame {

Once I added *private static final long serialVersionUID = 0*; the warning went away.



Project 4 was the final project in the SeaPort project series. It expanded on the first three projects by adding program functionality and greatly enhancing GUI appearance. More specifically, this project extended the multithreading capability implemented in Project 3 by considering the job *requirements*. The previously created `ArrayList persons`, containing a list of persons with particular skills at each port, was treated in this project as *resources pools*, along with supporting assignment to ships and jobs. Using these resource pools, Project 4 required that we demonstrate the concept of blocking the job threads established in Project 3 until required resources (i.e. dock and person with skills linked to job requirements) became available. I felt somewhat more comfortable starting this project having completed the multithreading functionality for Project 3. I continued to research ideas around multithreading, as well as different options for the GUI. Going into this project, I knew that I would likely be changing my GUI in several ways.

Similar to Project 3, I used a `synchronized` directive in my `run()` method within the Job class to avoid race conditions and ensure that a dock is only allowing one job to be performed for one ship at a time – i.e. the ship that is docked there. Syncing was carried out on a common element – the sea port, given that the port is still the only object that has the visibility/knowledge of the ships in queue and the docks located at that port. I kept the `JButton` (labeled 'Start') in my GUI to start the entire threading process, which is initiated by the user when they select this button. That way, the user has the option to search and sort the data structure before viewing the progress of the jobs. They can start the threading process whenever they choose. This does *not* allow the user to start each job manually, which would not meet the objective or requirements of multithreading for this project. Instead, this button just kicks off the entire process. The process must begin once the entire text file is read. Before a job begins, the `run()` method ensures that the ship for that job is docked and that there is no other job running for that ship, and then attempts to acquire the people it needs to fulfill its requirements. This is a new feature added to the multithreading process in Project 4. If these conditions are not met, it goes into a wait state using the `wait()` method. If the conditions are met, the job can begin, and the resources are claimed by that job. Once it is finished, it is synced to the sea port again before releasing the resources and sending a `notifyAll()` message to the other jobs in waiting. When a ship's jobs are all complete, that ship must leave the dock so that one from the queue can be docked there to begin its jobs. As with Project 3, the Thing field (parent) is used to link objects (e.g., linking a ship to its dock).

Similar to my approach for the first three projects, I used various strategies to test my code along the way. I continued to use print statements to check whether certain parts of the code were being reached. In addition, I paid special attention to what was taking place when I clicked

the ‘Status’ and ‘Cancel’ buttons. I wanted to be sure that when the ‘Status’ button was clicked, a job would be suspended yet would continue to hold onto its acquired resources (people), which would hold other jobs for that ship and other jobs that might need those resources from progressing. Furthermore, when I clicked the ‘Cancel’ button, I wanted to be sure that a job would be stopped and that it would release the resources. I watched where the resources (people) went, both when jobs were cancelled and when they were finished running. I checked both of the output tables under the ‘Job Progress’ tab to see that the resources (people) were displaying appropriately.

The most time-consuming aspect of Project 4 was likely getting the GUI to display all of the new capability that was implemented for the job threads. In particular, I found it difficult to implement two tables under the ‘Job Progress’ tab using the Swing class `JTable`. Previous to this project, I had minimal experience with this feature in Java. It was much more challenging to implement than I was expecting. I had trouble getting the ‘Status’ and ‘Cancel’ `JButtons` to function properly, as well as ensuring the information was being updated in the table. After struggling for a few days, I had decided to forgo using a `JTable` and go back to my original display of having all of the job threads in separate panels that made up one larger panel under the ‘Job Progress’ tab. I would have then added the additional information required here. However, I decided to try again to make the `JTable` work since I knew it would look much nicer. I am glad I did so. I learned that I needed to add a `MouseListener` class (`JTableButtonMouseListener`) for the `JButtons` because the JTable does not automatically forward a mouse click action on a cell to the component within that cell. The `MouseListener` class solves this by having the `mouseClicked()` method of the `MouseListener` call `doClick()` on the `JButton` component. This has the effect of

forwarding a mouse click on the `JTable` cell that contains the `JButton` to the `JButton` within that cell, thus triggering the `ActionListener` and subsequent event handler. This class is located within the World.java file. Other classes in the World file include `CustomContainerRenderer` and `TableModelClass,` which allowed me to implement the `JTables` properly with the `JButtons`.

I experimented with many new Swing features and updated my GUI in several ways, including those mentioned above. Most of these changes were to the 'Job Progress' tab of the output portion of the GUI. I utilized a `JSplitPane` to separate two `JTables` – one labeled 'Jobs' and the other labeled 'People'. The 'Jobs' table displays each job according to its port, ship, requirements, progress, status, and resources (people) being used. The 'People' table displays all of the people according to the port to which they belong, the skill they have, and the ship at which they are currently working (when applicable). The rest of the GUI remains unchanged with respect to appearance.

I believe I have gained a much better understanding on how to implement multithreading in a program while remaining cognizant of various issues that can occur when threads use shared resources, such as deadlock and starvation. The entire SeaPort project series truly challenged my object-oriented programming skills in many ways. It is likely the hardest project I have worked on at UMUC. Though it proved to be quite difficult at times, I believe its cumulative nature allowed me to learn much more than traditional projects. I have also enjoyed sharpening my GUI-creating skills by implementing and experimenting with different Swing features as I improved this program's GUI design throughout the weeks.

References:

Cordinc. (n.d.). *JButtons in a JTable.* Retrieved from https://www.cordinc.com/blog/2010/01/jbuttons-in-a-jtable.html.

Duchon, N. (2006, September 15). *Class Hierarchies – Notes.* Retrieved from http://sandsduchon.org/duchon/cs130/classes/ClassHierarchies.html.

Duchon, N. (n.d.). *Comparing Using Comparators.* Retrieved from http://sandsduchon.org/duchon/Musings/a/comparing.html.

Duchon, N. (2017, December 17.). *JTable Testing Framework.* Retrieved from http://sandsduchon.org/duchon/Musings/a/table.html.

Duchon, N. (n.d.). *JTextArea simple example with scrolling.* Retrieved from http://sandsduchon.org/duchon/Musings/a/textarea.html.

Duchon, N. (2013, August 6). *JTree and the Multi-Tree Data Structures.* Retrieved from http://sandsduchon.org/duchon/2014/sp/cs335/MusingsAll.html#JTree_and.

Duchon, N. (n.d.). *JTree Class Notes.* Retrieved from http://sandsduchon.org/duchon/Musings/a/tree.html.

Duchon, N. (2015, January 5). *Locks in Java: A Few Examples and Notes.* Retrieved from http://sandsduchon.org/duchon/cs335/threads/locks.html.

Duchon, N. (2017, March 26). *Reading a Text File Using Scanner.* Retrieved from http://sandsduchon.org/duchon/Musings/a/reading.html.

Duchon, N. (2016, July 10). *Project Philosophy – SeaPorts Project.* Retrieved from http://sandsduchon.org/duchon/2016/f/cs335/seaport.html#Suggestions:.

Duchon, N. (n.d.). *Scanner and JFileChooser.* Retrieved from http://sandsduchon.org/duchon/Musings/a/scanner.html.

Duchon, N. (n.d.). *Summary of Elementary Data Structures.* Retrieved from http://sandsduchon.org/duchon/Musings/a/summary.html.

Duchon, N. (2011, March 30). *Threads: A Few Examples.* Retrieved from http://sandsduchon.org/duchon/cs335/threads/index.html.

Duchon, N. (2017, April 5). *Treads 1.* Retrieved from http://sandsduchon.org/duchon/Musings/a/threads1.html.

Gobetz, W. (Photographer). (2003, February). *NYC: South Street Seaport - Pier 17 and Ambrose* [digital image]. Retrieved from https://www.flickr.com/photos/wallyg/153299820.

Marx, D. (2010, October 4). *How-To: Javac's -Xlint Options.* Retrieved from https://www.javaworld.com/article/2073587/javac-s--xlint-options.html.

Oracle (n.d.). *Class JTabbedPane.* Retrieved from https://docs.oracle.com/javase/8/docs/api/javax/swing/JTabbedPane.html.

Oracle (n.d.). *Class TableModelEvent.* Retrieved from https://docs.oracle.com/javase/8/docs/api/javax/swing/event/TableModelEvent.html.

Oracle (n.d.). *How to Use Borders.* Retrieved from https://docs.oracle.com/javase/tutorial/uiswing/components/border.html.

Oracle (n.d.). *How to Use BoxLayout.* Retrieved from https://docs.oracle.com/javase/tutorial/uiswing/layout/box.html.

Oracle (n.d.). *How to Use Split Panes.* Retrieved from https://docs.oracle.com/javase/tutorial/uiswing/components/splitpane.html#adding.

Oracle (n.d.). *How to Use Tabbed Panes.* Retrieved from https://docs.oracle.com/javase/tutorial/uiswing/components/tabbedpane.html.

Oracle (n.d.). *How to Use Tables.* Retrieved from https://docs.oracle.com/javase/tutorial/uiswing/components/table.html#modelchange.

Oracle (n.d.). *How to Write a Tree Selection Listener.* Retrieved from https://docs.oracle.com/javase/tutorial/uiswing/events/treeselectionlistener.html.

Oracle (n.d.). *Java Trail Code Sample – BoxLayoutDemo2.java.* Retrieved from https://docs.oracle.com/javase/tutorial/displayCode.html?code=https://docs.oracle.com/javase/tutorial/uiswing/examples/layout/BoxLayoutDemo2Project/src/layout/BoxLayoutDemo2.java.

StackOverflow (n.d.). *Making a JButton Clickable Inside a JTable.* Retrieved from https://stackoverflow.com/questions/10347983/making-a-jbutton-clickable-inside-a-jtable.

UCSB Engineering. (Photographer). (2006, September 22). *Computer program code* [digital image]. Retrieved from https://www.flickr.com/photos/sbengineer/3636620858.