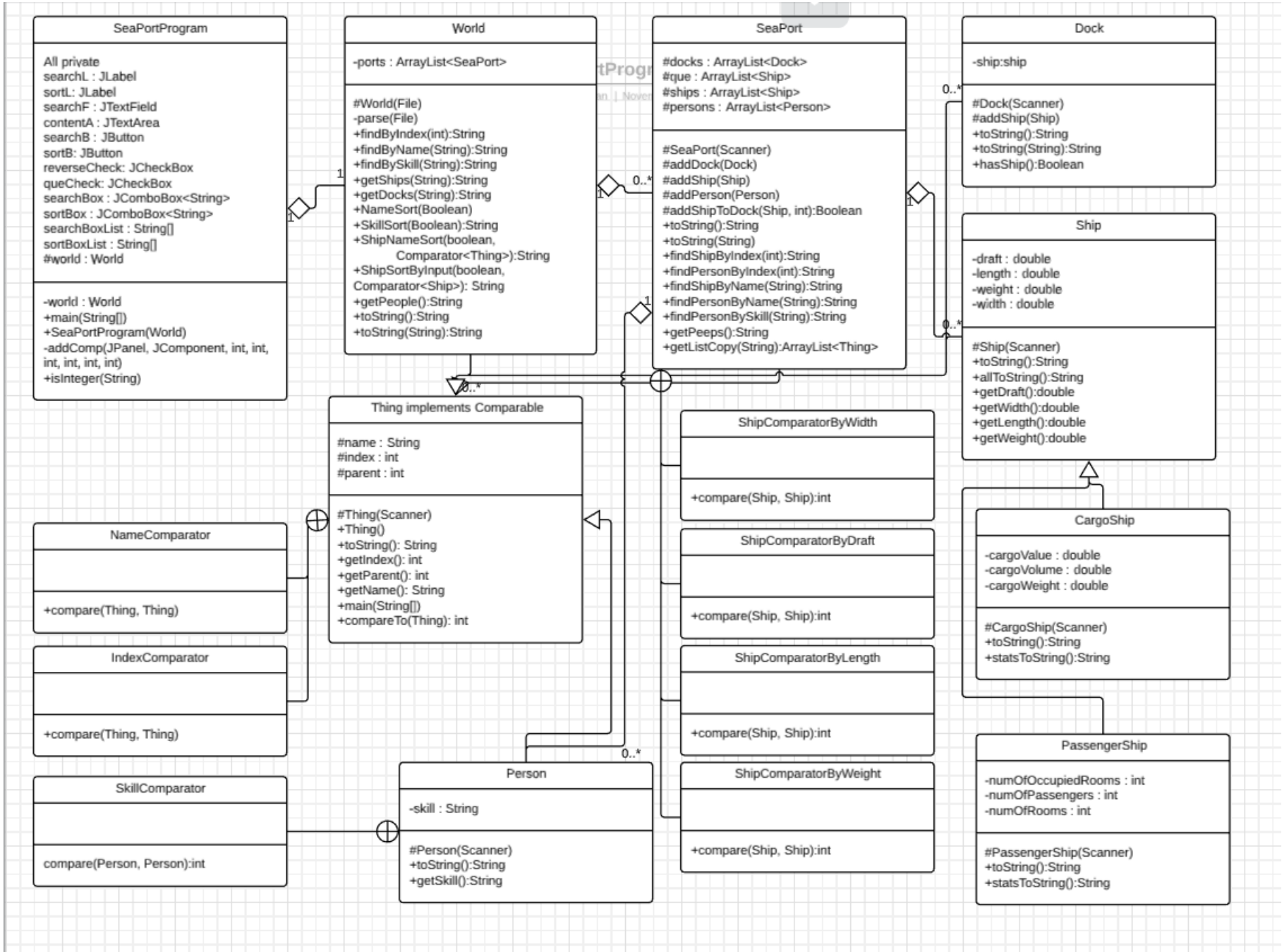
Zachary Finnegan

11/17/2019

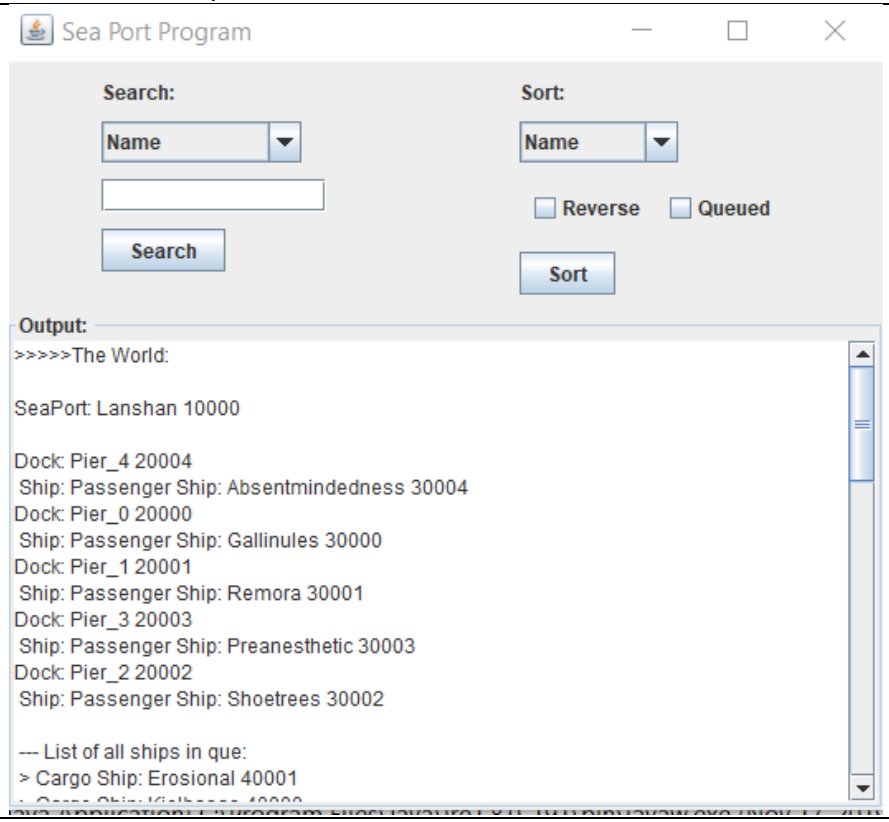
Project 2

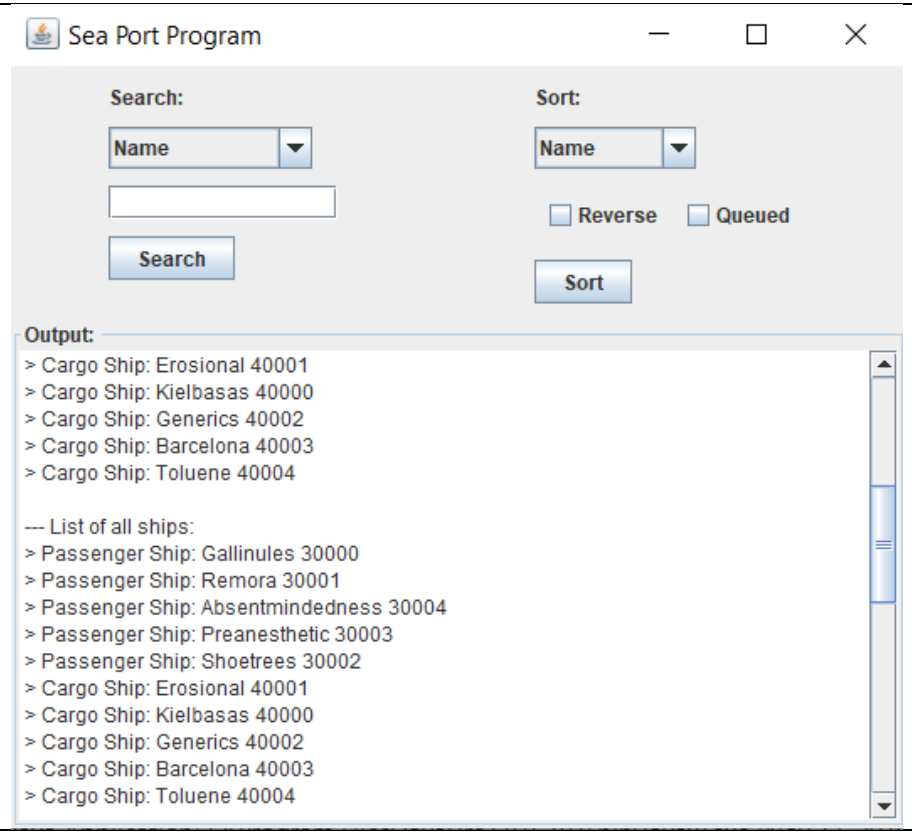
CMSC 335

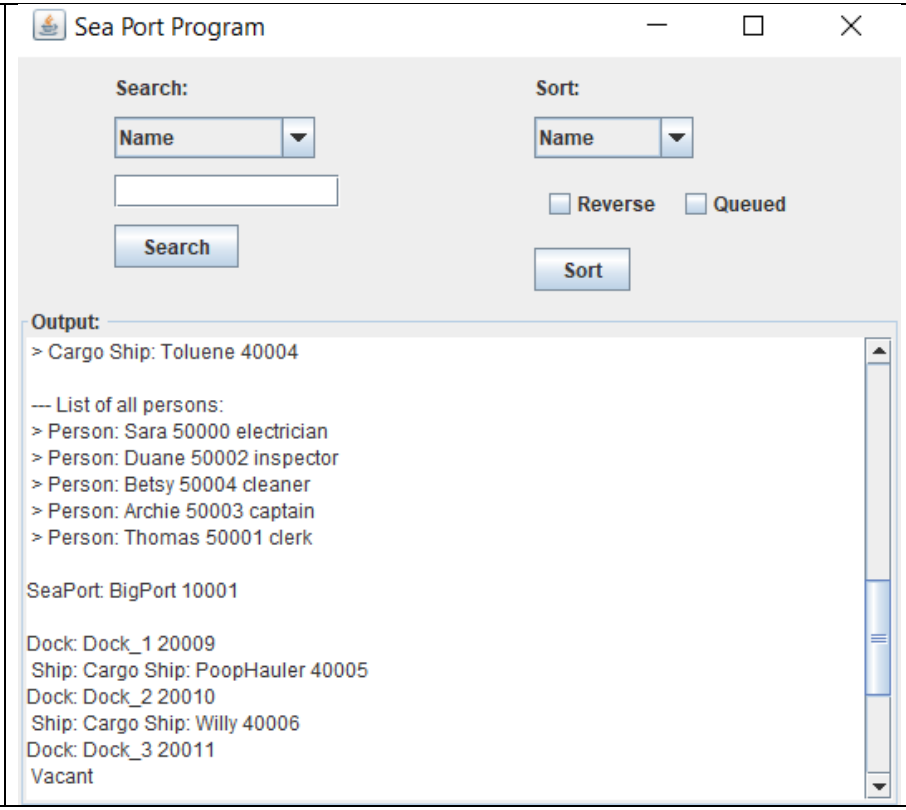
1. Design


SeaPortProgram
All private
searchL : JLabel
sortL: JLabel
searchF : JTextField
contentA : JTextArea
searchB : JButton
sortB: JButton
reverseCheck: JCheckBox
queCheck: JCheckBox
searchBox : JComboBox<String>
sortBox : JComboBox<String>
searchBoxList : String[]
sortBoxList : String[]
#world : World
-world : World
+main(String[])
+SeaPortProgram(World)
-addComp(JPanel, JComponent, int, int, int, int, int, int)
+isInteger(String)
World
-ports : ArrayList<SeaPort>
#World(File)
-parse(File)
+findByIndex(int):String
+findByName(String):String
+findBySkill(String):String
+getShips(String):String
+getDocks(String):String
+NameSort(Boolean)
+SkillSort(Boolean):String
+ShipNameSort(boolean, Comparator<Thing>):String
+ShipSortByInput(boolean, Comparator<Ship>): String
+getPeople():String
+toString():String
+toString(String):String
SeaPort
#docks : ArrayList<Dock>
#que : ArrayList<Ship>
#ships : ArrayList<Ship>
#persons : ArrayList<Person>
#SeaPort(Scanner)
#addDock(Dock)
#addShip(Ship)
#addPerson(Person)
#addShipToDock(Ship, int):Boolean
+toString():String
+toString(String)
+findShipByIndex(int):String
+findPersonByIndex(int):String
+findShipByName(String):String
+findPersonByName(String):String
+findPersonBySkill(String):String
+getPeeps():String
+getListCopy(String):ArrayList<Thing>
Dock
-ship:ship
#Dock(Scanner)
#addShip(Ship)
+toString():String
+toString(String):String
+hasShip():Boolean
Ship
-draft : double
-length : double
-weight : double
-width : double
#Ship(Scanner)
+toString():String
+allToString():String
+getDraft():double
+getWidth():double
+getLength():double
+getWeight():double
Thing implements Comparable
#name : String
#index : int
#parent : int
#Thing(Scanner)
+Thing()
+toString(): String
+getIndex(): int
+getParent(): int
+getName(): String
+main(String[])
+compareTo(Thing): int
ShipComparatorByWidth
+compare(Ship, Ship):int
ShipComparatorByDraft
+compare(Ship, Ship):int
ShipComparatorByLength
+compare(Ship, Ship):int
ShipComparatorByWeight
+compare(Ship, Ship):int
NameComparator
+compare(Thing, Thing)
IndexComparator
+compare(Thing, Thing)
SkillComparator
compare(Person, Person):int
Person
-skill : String
#Person(Scanner)
+toString():String
+getSkill():String
CargoShip
-cargoValue : double
-cargoVolume : double
-cargoWeight : double
#CargoShip(Scanner)
+toString():String
+statsToString():String
PassengerShip
-numOfOccupiedRooms : int
-numOfPassengers : int
-numOfRooms : int
#PassengerShip(Scanner)
+toString():String
+statsToString():String
1
0..*

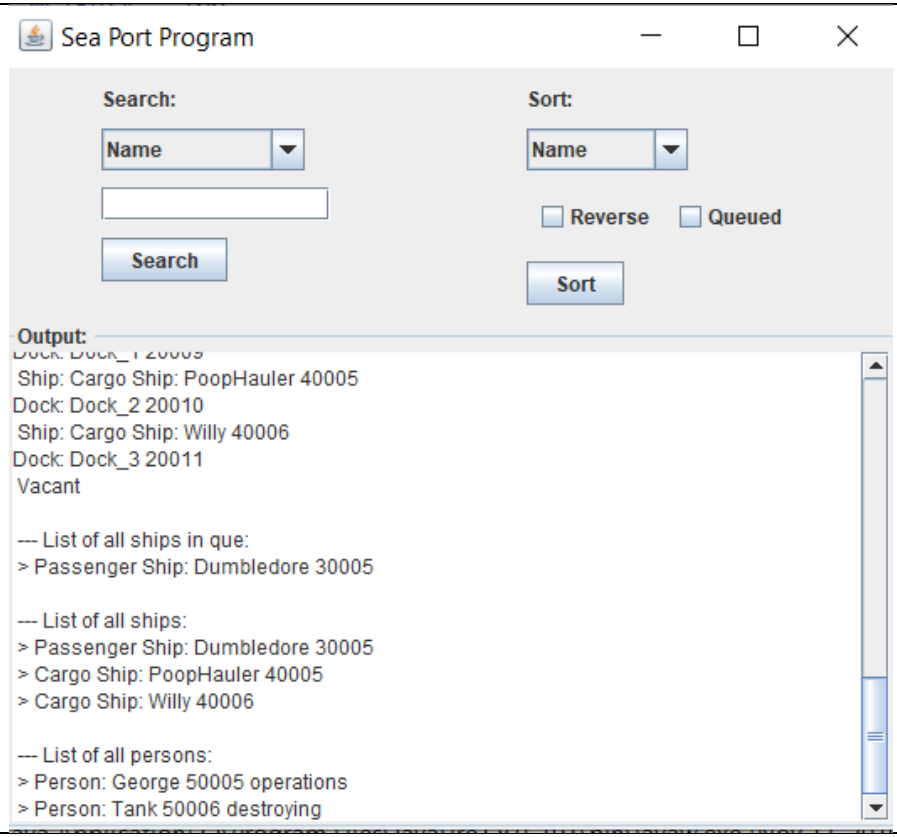
2. User's Guide
    a. Add the package to your IDE path. Import it to your IDE and then press run. Select the SeaPortData.txt when the JFileChooser pops up and then search and sort to your hearts content.
3. Test Plan
    a. This table will only represent new functionality. See previous documentation to see test cases for the original program.

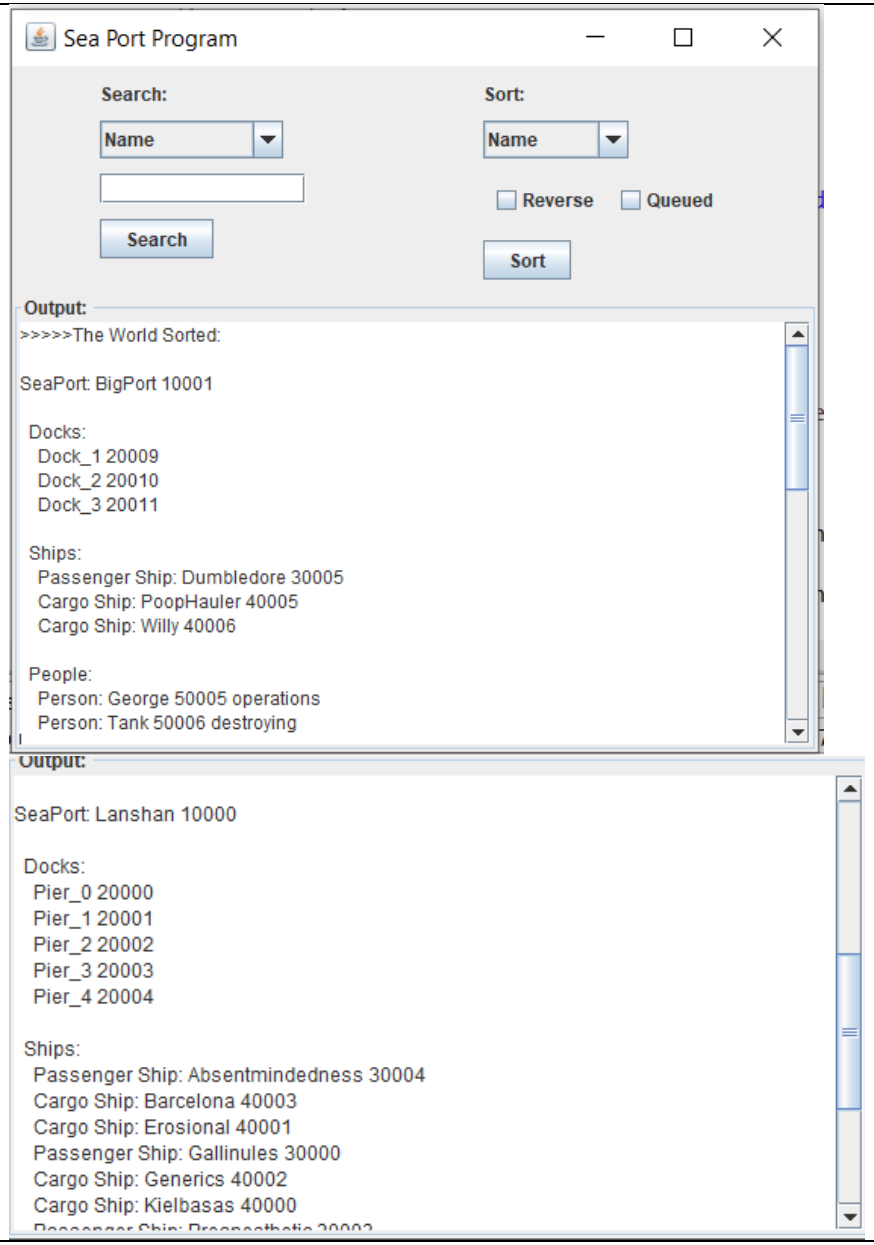
Test	Input	Expected Output	Screenshot Output
Select a file and load, parse and display it	SeaPortData.txt	The updated GUI. With the same output as the previous iteration of the program.	

			 Sea Port Program Search: Name Search Sort: Name Reverse Queued Sort Output: > Cargo Ship: Erosional 40001 > Cargo Ship: Kielbasas 40000 > Cargo Ship: Generics 40002 > Cargo Ship: Barcelona 40003 > Cargo Ship: Toluene 40004 --- List of all ships: > Passenger Ship: Gallinules 30000 > Passenger Ship: Remora 30001 > Passenger Ship: Absentmindedness 30004 > Passenger Ship: Preanesthetic 30003 > Passenger Ship: Shoetrees 30002 > Cargo Ship: Erosional 40001 > Cargo Ship: Kielbasas 40000 > Cargo Ship: Generics 40002 > Cargo Ship: Barcelona 40003 > Cargo Ship: Toluene 40004

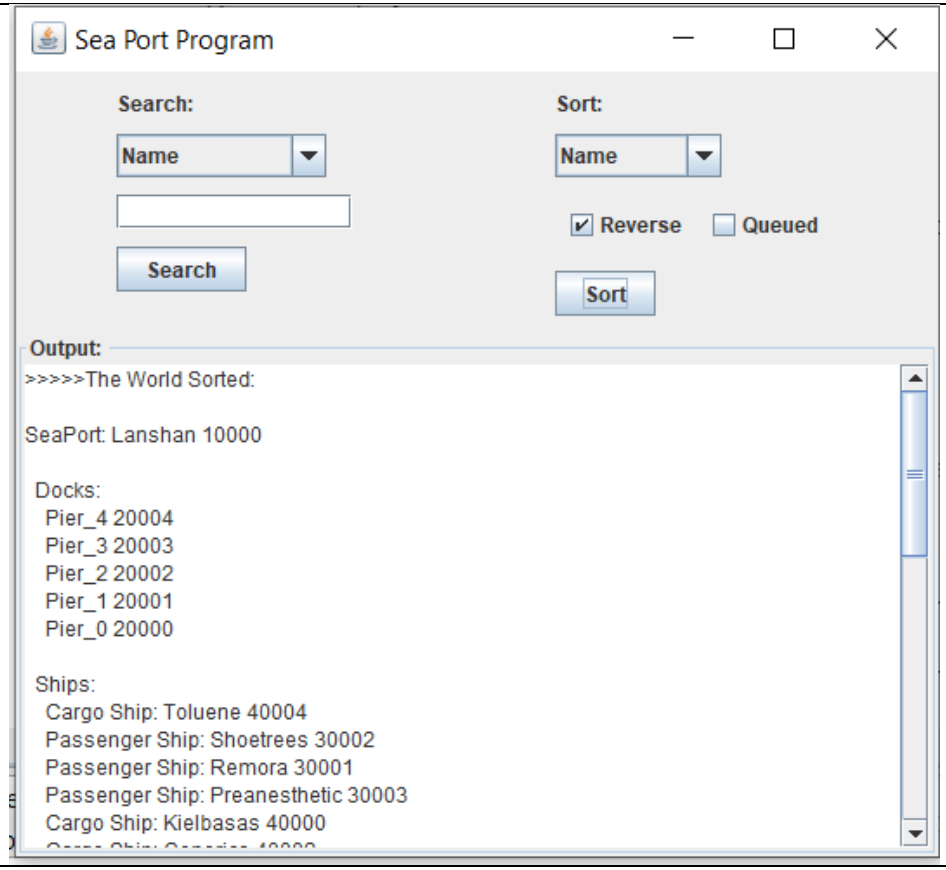
			

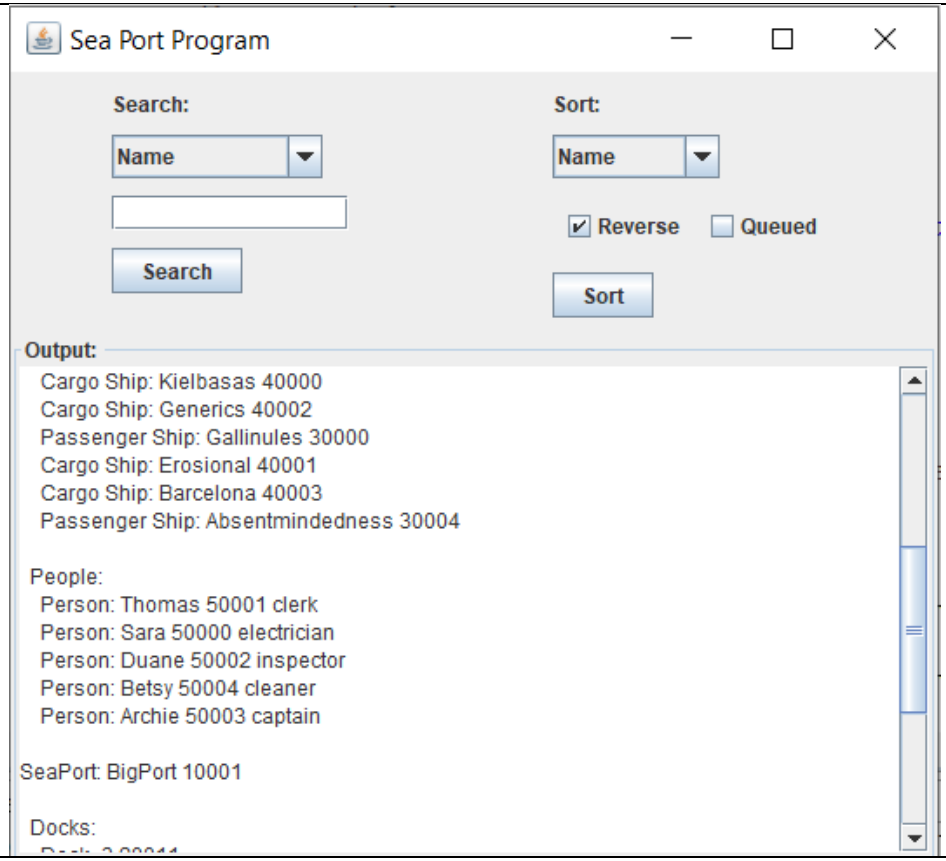
Sea Port Program
Search:
Name
Search
Sort:
Name
Reverse
Queued
Sort
Output:
> Cargo Ship: Toluene 40004
--- List of all persons:
> Person: Sara 50000 electrician
> Person: Duane 50002 inspector
> Person: Betsy 50004 cleaner
> Person: Archie 50003 captain
> Person: Thomas 50001 clerk
SeaPort: BigPort 10001
Dock: Dock_1 20009
Ship: Cargo Ship: PoopHauler 40005
Dock: Dock_2 20010
Ship: Cargo Ship: Willy 40006
Dock: Dock_3 20011
Vacant

			 Sea Port Program Search: Name Search Sort: Name Reverse Queued Sort Output: Ship: Cargo Ship: PoopHauler 40005 Dock: Dock_2 20010 Ship: Cargo Ship: Willy 40006 Dock: Dock_3 20011 Vacant --- List of all ships in que: > Passenger Ship: Dumbledore 30005 --- List of all ships: > Passenger Ship: Dumbledore 30005 > Cargo Ship: PoopHauler 40005 > Cargo Ship: Willy 40006 --- List of all persons: > Person: George 50005 operations > Person: Tank 50006 destroying

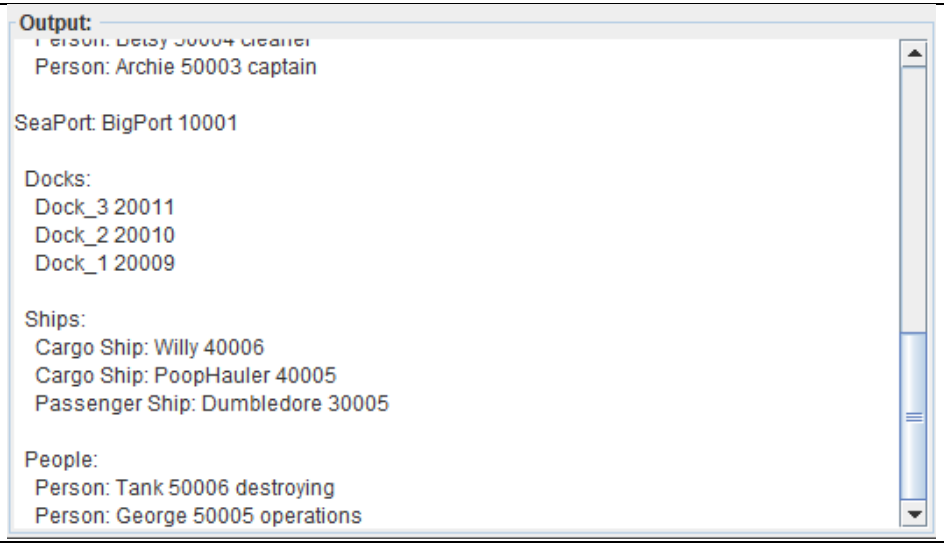
Sort all objects by name normal and reverse order	Sort ComboBox = Name Reverse CheckBox = Unchecked	Ports, docks, ships and people will be listed in alphabetical order by name. Port: Docks: A-Z… Ships: A-Z… People: A-Z…	

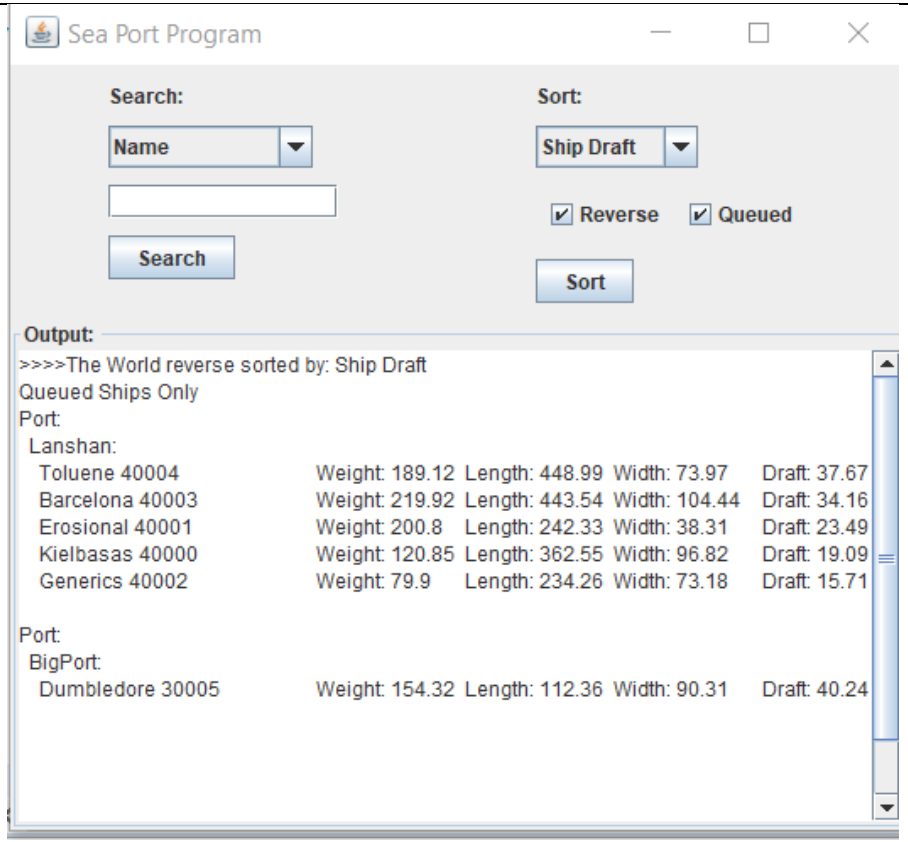
			Output: Ships: Passenger Ship: Absentmindedness 30004 Cargo Ship: Barcelona 40003 Cargo Ship: Erosional 40001 Passenger Ship: Gallinules 30000 Cargo Ship: Generics 40002 Cargo Ship: Kielbasas 40000 Passenger Ship: Preanesthetic 30003 Passenger Ship: Remora 30001 Passenger Ship: Shoetrees 30002 Cargo Ship: Toluene 40004 People: Person: Archie 50003 captain Person: Betsy 50004 cleaner Person: Duane 50002 inspector Person: Sara 50000 electrician Person: Thomas 50001 clerk

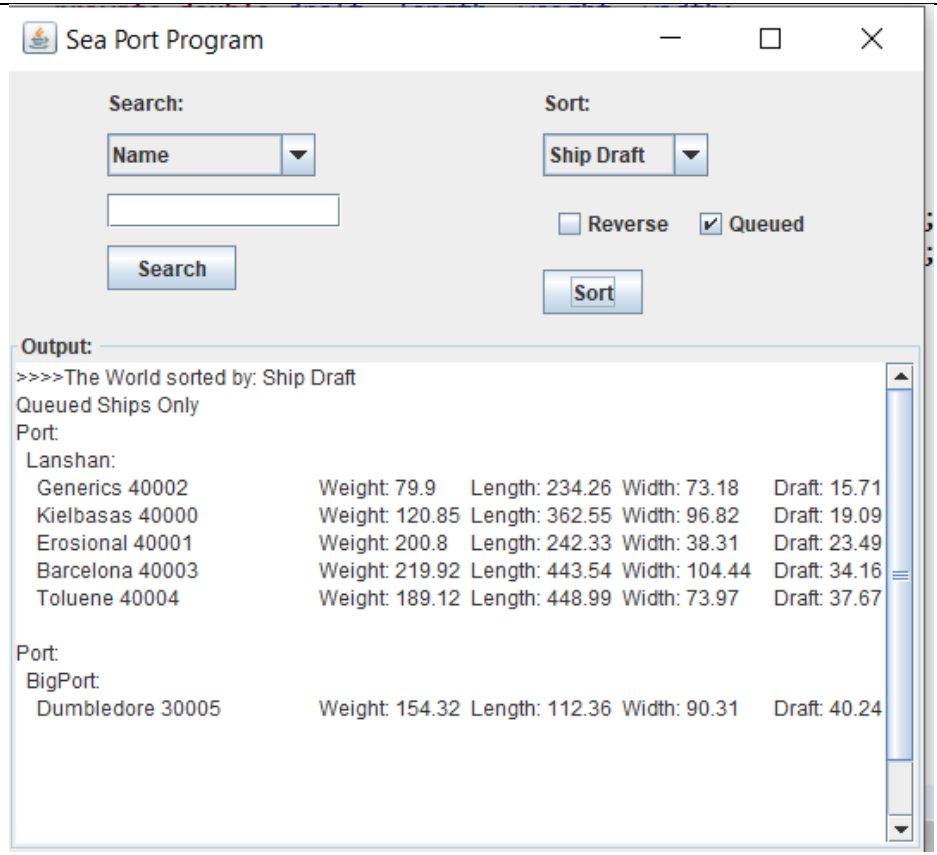
	Sort ComboBox = Name Reverse CheckBox = Checked	Same as previous but with ports and all objects in reverse alphabetical order	

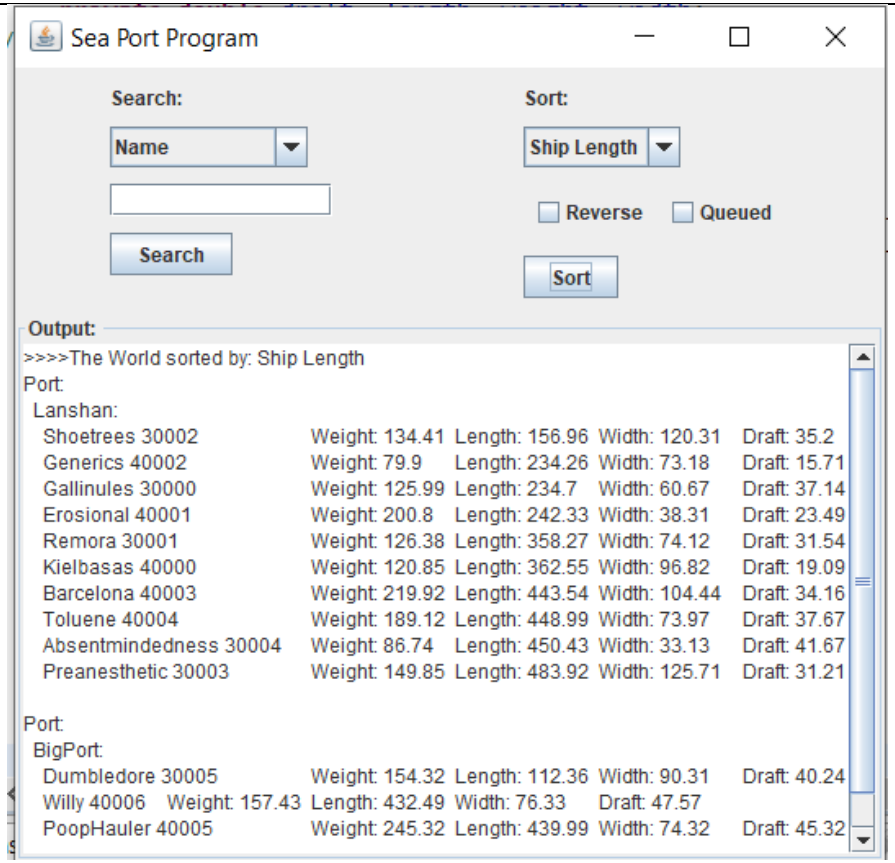
			

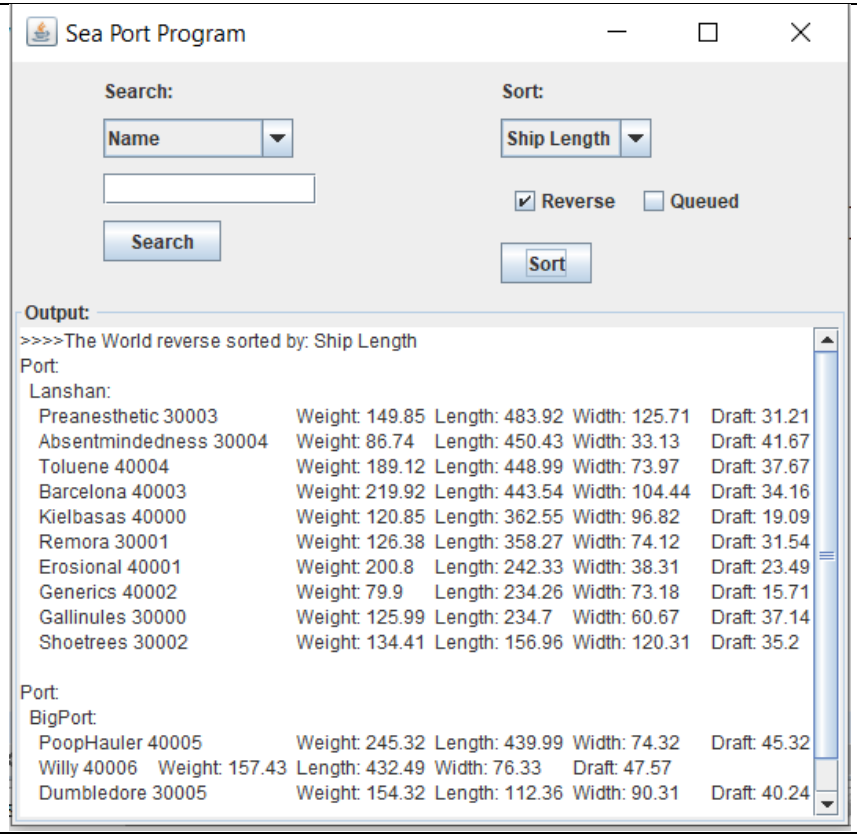
Sea Port Program
Search:
Name
Search
Sort:
Name
Reverse
Queued
Sort
Output:
Cargo Ship: Kielbasas 40000
Cargo Ship: Generics 40002
Passenger Ship: Gallinules 30000
Cargo Ship: Erosional 40001
Cargo Ship: Barcelona 40003
Passenger Ship: Absentmindedness 30004
People:
Person: Thomas 50001 clerk
Person: Sara 50000 electrician
Person: Duane 50002 inspector
Person: Betsy 50004 cleaner
Person: Archie 50003 captain
SeaPort: BigPort 10001
Docks:

			 Output: Person: Archie 50003 captain SeaPort: BigPort 10001 Docks: Dock_3 20011 Dock_2 20010 Dock_1 20009 Ships: Cargo Ship: Willy 40006 Cargo Ship: PoopHauler 40005 Passenger Ship: Dumbledore 30005 People: Person: Tank 50006 destroying Person: George 50005 operations

Sort queued ships by Draft in low to high and high to low order	Sort ComboBox = Ship Draft Reverse CheckBox = Checked Que CheckBox = Checked	All ships in ques at all ports with draft high to low within their respective ports.	

	Sort ComboBox = Ship Draft Reverse CheckBox = Unchecked Que CheckBox = Checked	All ships in ques at all ports with draft low to high within their respective ports.	

Sort all ships by length high to low and low to high	Sort ComboBox = Ship Length Reverse CheckBox = Unchecked Que CheckBox = unchecked	All ships sorted by length low to high within their respective ports.	

	Sort ComboBox = Ship Length Reverse CheckBox = Checked Que CheckBox = unchecked	All ships sorted by length high to low within their respective ports.	

4. Lessons Learned

   I updated the GUI to remove all the radio buttons. Frankly it was a little much in the previous iteration and if I had used them in this iteration of the program it would have gotten out of control. Instead I implemented ComboBoxes and a couple of CheckBoxes. If I add additional sub sorting I will either add an additional ComboBox in the sort section of the GUI or I will add more options in the already existing sort ComboBox.

I am pretty happy with the GUI. I don't love how the CheckBoxes sit in the Sort panel. Next time I am going to implement the JTable for displaying the ships and their specifics. I tried putting tabs between each stat and that helped a lot but it still isn't great when displayed in the textarea.

I also implemented lambda action listeners to the buttons which kind of complicated the GUI code and added a lot of extra lines even after I streamlined the calls within in switch case. Ideally That would occur in a different area than the creation of the look and components of the GUI. Maybe there is a better way to do it.

I changed all ArrayLists to protected from private so that I could get better access to them from other classes in the package. I am not sure if this follows best practices, but it makes the code a lot cleaner with fewer calls and methods. Speaking of which I found a really compact way to sort the lists and return the sorted string. I am extremely proud of the "ShipSortByInput" method in the World class. At first it was probably 50 lines of code. I found this extremely obnoxious and it was a lot of repeating code. I tried various different approaches until I settled on creating the comparator object in the SeaPortProgram and sending that along with the Boolean value of the que checkbox in the call to the "ShipSortByInput" method. That along with a temporary ArrayList to hold the que or ships ArrayList allowed for a lot of versatility in 17 lines of code.

The Comparator classes I created are kind of spread out through the classes in the package. I am wondering if it might be better to create a general comparator class with different static methods inside of it. Any feedback on that idea or any of the other things I did would be appreciated.