SeaPort Project: 4

Jeffrey Killen

12/15/19

CMSC 335 7980

University of Maryland Global Campus

Table of Contents

Design:

UML Diagram ………………………………………………………………………...….4

Summary of Design Elements ……………………………………………………………5

User’s Guide:

Usage ……………………………………………………………………………………..7

Test Plan:

Open file ………………………………………………………………………………….9

Search ……………………………………………………………………………………11

Sorting……………………………………………………………………………………14

Job Completion…………………………………………………………………………..18

Lessons Learned:

General………..………………………………………………………………………….19

SeaPort Project: 1

**Design:**

**Problem Statement:**

The user must select a File to open. This File is used to create a Scanner. The Scanner is used to create a World. The World builds a Tree with the Scanner. The World displays the Tree to the user. The user enters a value to search. World searches the Tree. World displays the Results to the user. The user selects a sort criteria. World sorts the selected ArrayList by the select value. World displays the Results to the user.

As Jobs are created, they must get their Ship to dock, acquire workers from the SeaPort and perform some work.

GUI:

* User must select a File to open.
* This File is used to create a Scanner.
* The Scanner is used to create a World.
* Display String to the user
* User enters a value
* User selects a sort criteria
* User clicks a button

World:

* Build a Tree with the Scanner
* Search the Tree
* Sort the tree

<Thing>:

* Comparable
* Iterable (ArrayList<>)
* Will be searched
* toString()
* Can be sorted

Job:

* Runnable
* Requires Resources (Dock, and some Persons)
* Acquire Dock from an ArrayBlockingQueue
* Acquire Persons from a ConcurrentHashMap

**UML Diagram:**

A close up of text on a white surface

Description automatically generated

Figure 1: Class Diagram

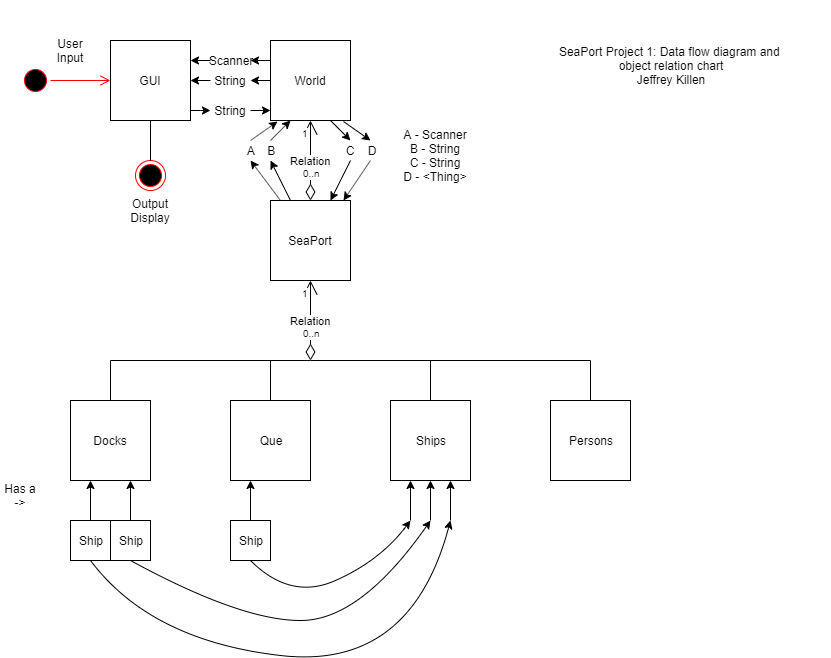


Figure 2: Data Flow and Structure

**Summary of elements:**

>SeaPortProgram extends JFrame-

This Class has the main function of the program. This class serves as the GUI for the program. The user can interact with a button to open a File using JFileChooser. This File is used to create a Scanner. The Scanner is used to create a World. The user can enter text or a value into a JTextField and interact with a JButton prompting World to perform a search. A String can be displayed in a JTextArea. This class performs all Exception handling for the program.

>PortTime

This Class is a getter/setter wrapper for an int.

>Thing extends Comparable

This Class serves as a Parent to several other data structures. This Class provides the basic searchable data and methods to construct, get, compare, and display that data.

>World extends Thing

This Class is a Child of Thing. This Class contains the methods to build and search a tree of <Thing>. This Class can get a <Thing> by its index and assign another <Thing> to it. The toString() method is overridden to start a cascading toSting() call to the entire tree.

This class also contains a method to sort Ships, Docks, Seaports, and Persons.

>SeaPort extends Thing

This Class contains data, getters, and overridden compareTo() and toString() methods.

>Dock extends Thing

This Class contains data, getters, and overridden compareTo() and toString() methods. This Class also contains a Ship.

>Ship extends Thing

This Class serves as a Parent to other data structures. This Class provides the basic searchable data and overridden methods to compare, and display that data.

>PassengerShip extends Ship

This Class contains data, getters, and overridden compareTo() and toString() methods.

>CargoShip extends Ship

This Class contains data, getters, and overridden compareTo() and toString() methods.

>Person extends Thing

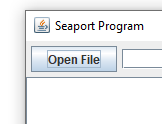
This Class contains data, getters, and overridden compareTo() and toString() methods. Each person has a skill and can be used by a Job to do some work.

>Job extends Thing implements Runnable

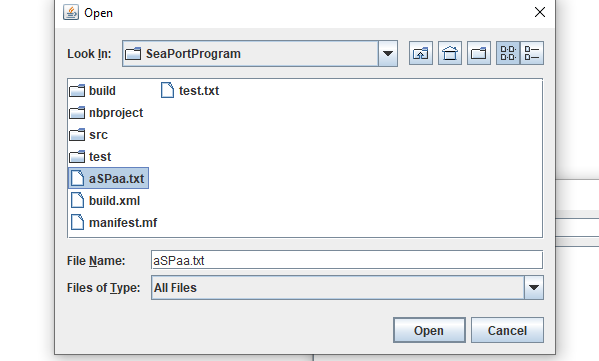
This class serves as a Thread for simulating some work to be performed. Work is only performed when the ship that owns this Job is docked and the workers with required skills are available at the SeaPort.

**User Guide:**

1. Click button labeled “Open File”



1. Select File to open (e.g. aSPaa.txt)
2. Click Open button



1. Type a value to search in the Text Field
2. Click the button labeled “Search”

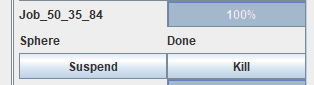


1. Select a sort criteria and click the sort button

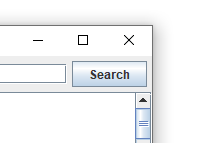
A screenshot of a cell phone

Description automatically generated

1. Click the Suspend/Kill button to Suspend/Kill the Thread.



1. Click the “X” to exit the program

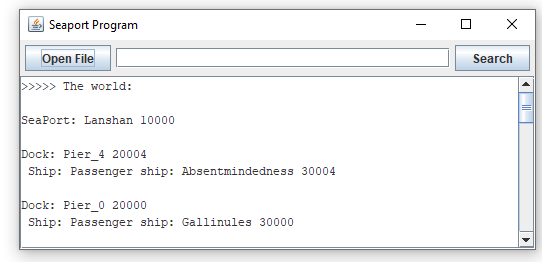


**Test Plan:**

Open File – Valid File

* Click the “Open File” Button
* Select “aSPaa.txt”
* Click the “Open” Button

Expected output – Text display, contents of the world:

Actual output - 

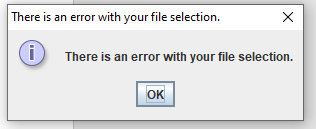
Result – pass

Open File – Invalid File

* Click the “Open File” Button
* Select “test.txt”
* Click the “Open” Button

Expected output – Error Message

Actual output –



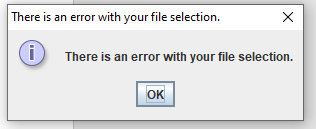
Result – pass

Open file – Cancel opening

* Click the “Open File” Button
* Click the “Cancel” Button

Expected output – error message

Actual output –



Result – pass

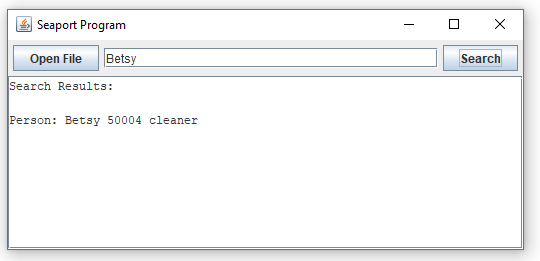
Search World – Valid Search Term

* Type “Betsy” into the text field
* Click the “Search” Button

Expected output –

Search Results:

Person: Betsy 500004 cleaner

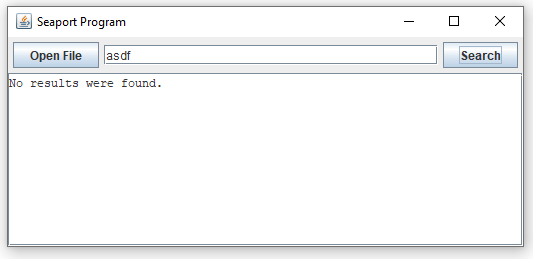
Actual output - 

Result – pass

Search World – Invalid input

* Type “asdf” into text field
* Click the “Search” Button

Expected output – “No results were found.”

Actual output - 

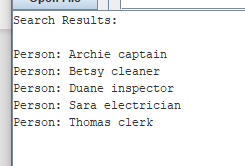
Result – pass

Sort - Persons By Name

* Select “Persons By Name”
* Click the sort Button

Expected Output – list of all persons sorted alphabetically by name.

Actual Output –



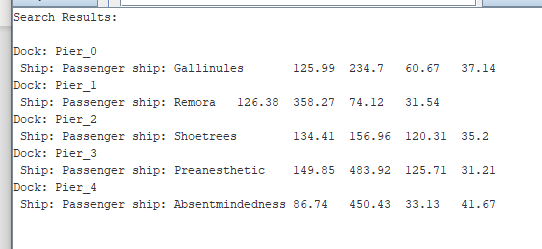
Result – pass

Sort – Docks By Name

* Select “Docks By Name”
* Click the sort Button

Expected Output – list of all docks sorted alphabetically by name.

Actual Output –



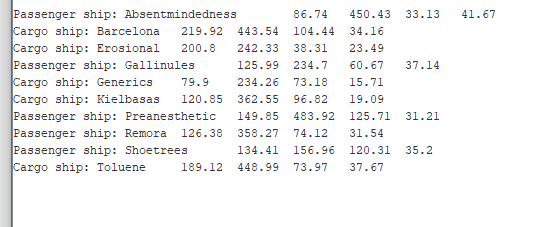
Result – pass

Sort – All Ships By Name

* Select “All Ships By Name”
* Click the sort button

Expected Output – A list of all ships sorted alphabetically by name.

Actual Output –



Result – pass

Sort – Ship/Que By Weight/Length/Width/Draft

* Select “All Ships” OR ”Que” By “Weight” OR “ Length” OR “Width” OR “Draft”
* Click the sort button

Expected Output – a list of Selected Types sorted by the Selected Value

Actual Outputs –

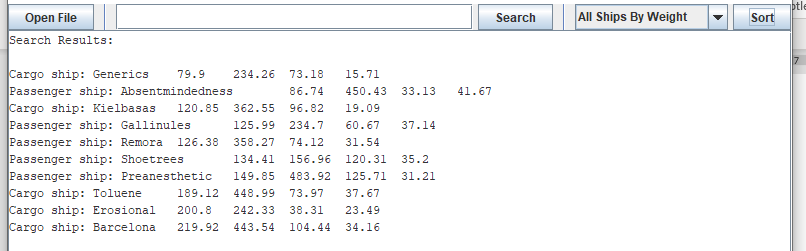


Figure All Ships By Weight

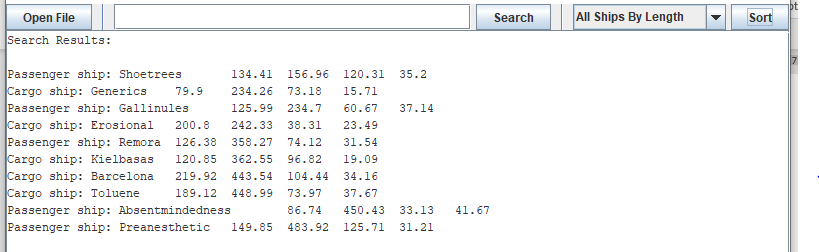


Figure All Ship By Length

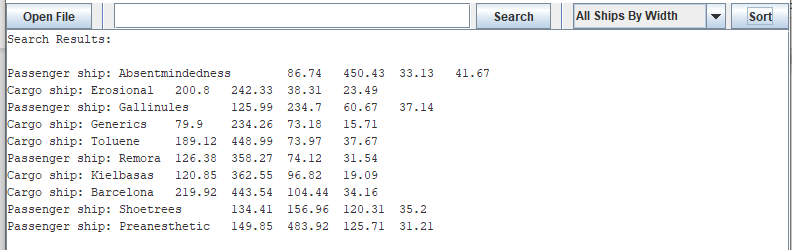


Figure All Ships By Width

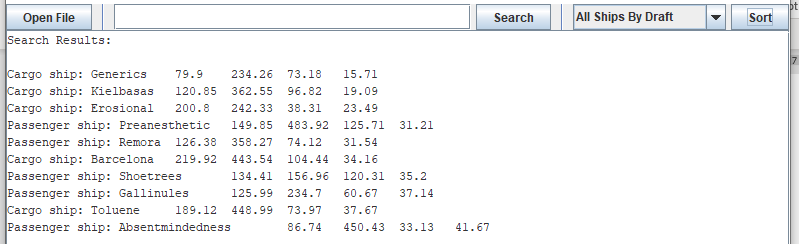


Figure All Ships By Draft

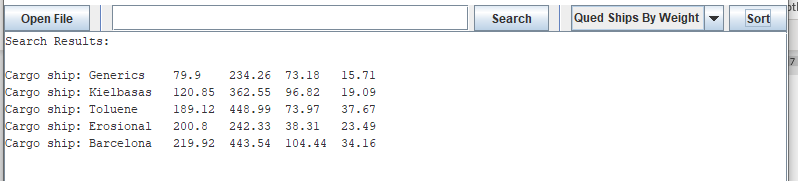


Figure Qued Ships By Weight

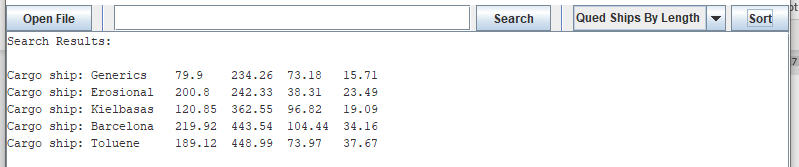


Figure Qued Ship By Length

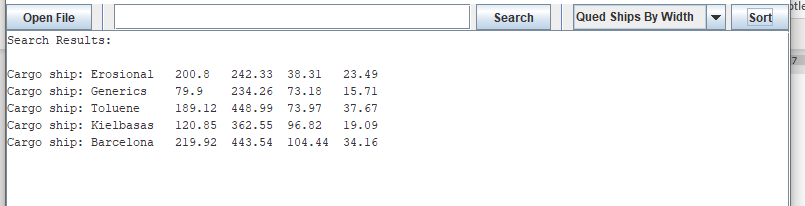


Figure Qued Ships By Width

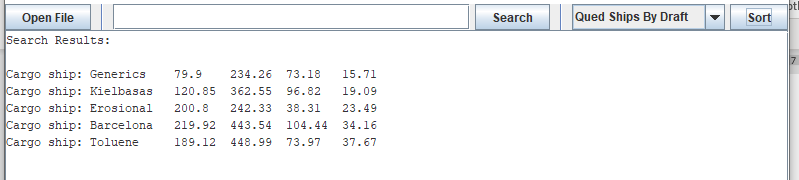


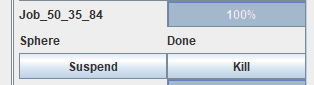
Figure Qued Ships By Draft

Result – pass

Suspend/Kill Thread

* Click Suspend/Kill button for Thread

Expected output – Thread status update to suspend/done

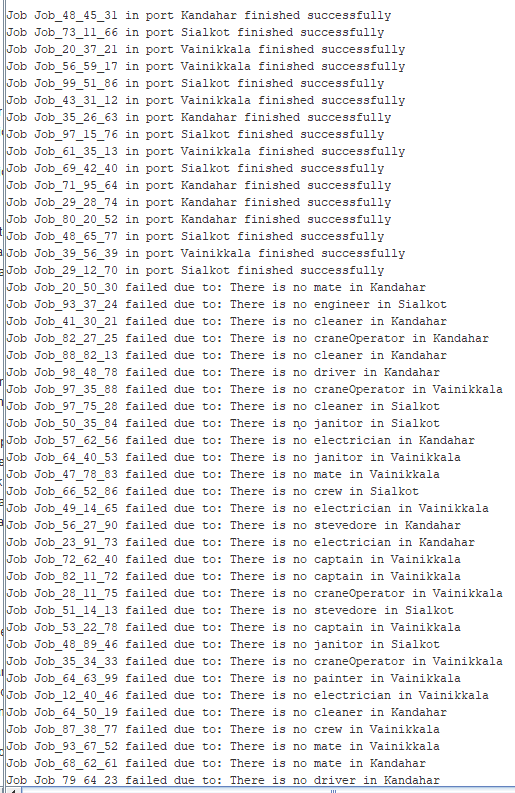


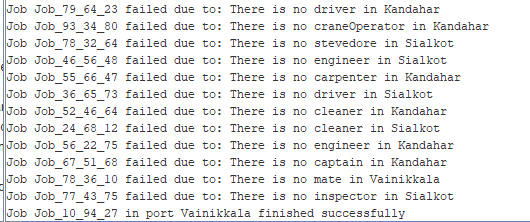
Result – pass

Perform all Jobs

* Open file aSPad.txt

Expected output – 17 Jobs complete successfully, the rest fail due to lack of workers





Result - pass

**Lessons Learned:**

* Even if you are stuck in bed with a cold, take the time to transcribe your notes to the documentation. Doing it at the last minute is not fun.
* Don’t overthink the efficiency of an algorithm that is guaranteed to be replaced later.
* Design Exception handling as you would any other element of the program. It is not an afterthought.
* Check your design against the next Project to make sure you are not doubling your work. I have a feeling that will be the case with Project 2.
* Lambda Expressions.
* Build test programs to learn new tools.
* Pay attention to details
* Sometimes life gets in the way, just do your best and don’t beat yourself up.
* Don’t transfer at work and move during the semester.
* Concurrency is a deep rabbit hole, spend some time with it
* I need more practice with generics, concurrency, and GUIs.