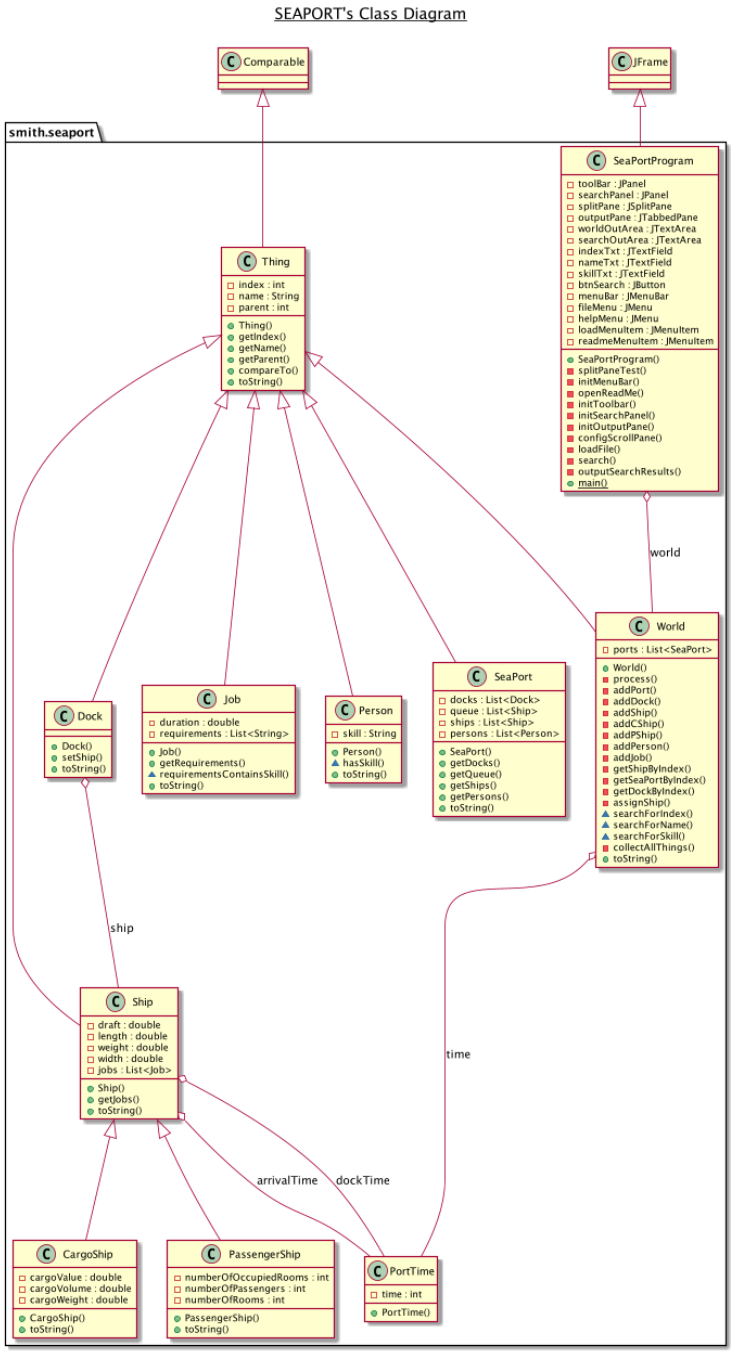
Hunter T. Smith
March 31, 2019
CMSC 335 - Object-Oriented and Concurrent Programming
Project: 1

Overview

A program that simulate some of the aspects of a number of Sea Ports. Providing experience with polymorphism, inheritance, and OOP styling.

Design

UML Diagram:



Explanations:

The included JavaDoc contains a thorough explanation of the classes and many of the methods (private methods are not shown in the JavaDoc, but are commented the same in source code); However, Summarizing some of the aspects and requirements of this project here is beneficial, but is meant to supplement the information given by the included JavaDoc.

The Driver of the program is the SeaPortProgram class. The class creates and handles the GUI and user inputs. The SeaportProgram class communicates with other classes in the program through a it's class field, world, an instance of the World class. After instantiating a World, The main methods used to interact with it is the search method. The search method reads the users search parameters and iterates through the worlds SeaPorts to find matches fulfilling the primary requirement of this project.

All other classes except PortTime, which is currently not being used, are a subclass of the Thing. The Thing class enables polymorphic behavior in this project. Throughout the project the subclasses are accessed through Lists of type Thing but with different results due to overriding methods (e.g., toString()). Within the subclasses of Thing, the SeaPort classes is one of the most important from a hierarchal search view.

The Seaport Class represents an instance of a simulated seaport. As mentioned earlier, all SeaPorts are included in a list within an instance of the World class. Seaports contain lists of all docks, ships (which hold Jobs), and persons (which have Skills) at the current instance with getter methods. This provides the basis for the hierarchal search data structure and enables the World Class the ability to perform searches by index, name, and skill for subclasses of Thing; thus fulfilling the project requirements.

User Guide

A comprehensive user guide is provided in the Seaport Program folder (./src/res/SeaPortProgram_UserManual.pdf).

Test Plan

ID	Selected Input	Expected output	Actual Output (Screenshots)	Pass/Fail
1	click help menu and open program instructions	pdf opens in systems configured pdf viewer.	ScreenShots: fig.1.0	Pass
2	Search with no file loaded	Display error message	ScreenShots: fig.1.1	Pass

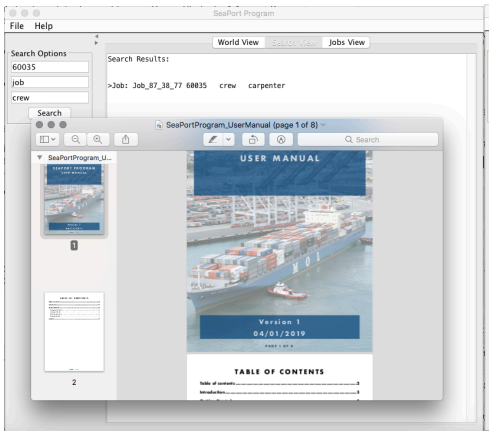
3	Click file menu, load file	Display JFileChooser	ScreenShots: fig.1.2	Pass
4	load file aSPaa.txt	File loads and displays world neatly formatted in the world viewer.	ScreenShots: fig.2.0	Pass

5	name search: “pier”	Switches to search Viewer and displays:
		Search Results:
		
		>Dock: Pier_4 20004
		  Ship: Passenger ship: Absentmindedness 30004
		>Dock: Pier_0 20000
		  Ship: Passenger ship: Gallinules 30000
		>Dock: Pier_1 20001
		  Ship: Passenger ship: Remora 30001
		>Dock: Pier_3 20003
		  Ship: Passenger ship: Preanesthetic 30003
		>Dock: Pier_2 20002
		  Ship: Passenger ship: Shoetrees 30002
			ScreenShots: fig.2.1	Pass

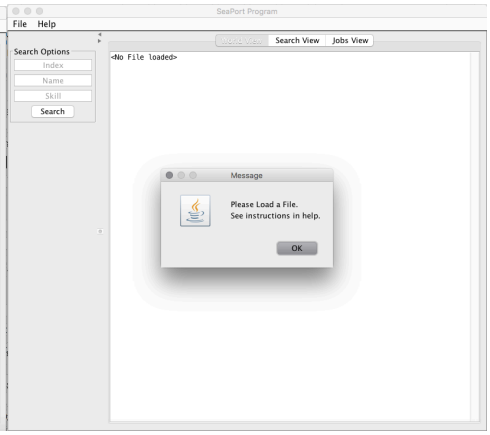
6	index search: “20001”	displays: Search Results: >Dock: Pier_1 20001 Ship: Passenger ship: Remora 30001	ScreenShots: fig.2.2	Pass
7	skill search: “cap”	displays: Search Results: >Person: Archie 50003 captain	ScreenShots: fig.2.3	Pass
8	load file aSPad.txt	File loads and displays world neatly formatted in the world viewer.	ScreenShots: fig.3.0	Pass

9	name search: “job”	Switches to search Viewer and displays: Search Results: >Job: Job_99_51_86 60002 driver >Job: Job_73_11_66 60004 >Job: Job_48_65_77 60003 electrician driver >Job: Job_48_89_46 60000 janitor >Job: Job_97_15_76 60001 driver >Job: Job_97_75_28 60009 cleaner painter janitor >Job: Job_51_14_13 60008 stevedore >Job: Job_36_65_73 60006 driver driver >Job: Job_46_56_48 60005 engineer >Job: Job_29_12_70 60007 painter >Job: Job_66_52_86 60015 crew >Job: Job_78_32_64 60014 mate clerk stevedore carpenter >Job: Job_77_43_75 60013 electrician inspector >Job: Job_24_68_12 60016 mate cleaner >Job: Job_50_35_84 60011 janitor driver captain >Job: Job_93_37_24 60012 engineer crew inspector >Job: Job_69_42_40 60010 clerk >Job: Job_41_30_21 60040 cleaner >Job: Job_23_91_73 60039 painter electrician painter craneOperator >Job: Job_56_22_75 60038 engineer electrician >Job: Job_67_51_68 60037 captain >Job: Job_55_66_47 60045 carpenter captain cleaner >Job: Job_79_64_23 60046 inspector driver >Job: Job_52_46_64 60044 clerk inspector cleaner >Job: Job_82_27_25 60041 craneOperator >Job: Job_98_48_78 60043 driver clerk mate >Job: Job_80_20_52 60042 crew >Job: Job_20_50_30 60057 mate >Job: Job_48_45_31 60047 >Job: Job_64_50_19 60048 painter cleaner >Job: Job_35_26_63 60049 painter >Job: Job_29_28_74 60050 >Job: Job_56_27_90 60056 stevedore captain carpenter >Job: Job_57_62_56 60055 electrician carpenter >Job: Job_88_82_13 60054 cleaner electrician >Job: Job_71_95_64 60052 >Job: Job_68_62_61 60053 mate captain stevedore >Job: Job_93_34_80 60051 mechanic craneOperator >Job: Job_10_94_27 60020 carpenter cleaner clerk >Job: Job_20_37_21 60021 >Job: Job_64_63_99 60022 painter >Job: Job_93_67_52 60019 mate >Job: Job_43_31_12 60018 inspector >Job: Job_61_35_13 60017 >Job: Job_64_40_53 60034 janitor >Job: Job_87_38_77 60035 crew carpenter >Job: Job_53_22_78 60032 captain stevedore >Job: Job_12_40_46 60033 electrician carpenter captain >Job: Job_97_35_88 60030 craneOperator >Job: Job_72_62_40 60028 captain >Job: Job_49_14_65 60027 inspector electrician electrician >Job: Job_56_59_17 60029 cleaner >Job: Job_35_34_33 60031 craneOperator >Job: Job_47_78_83 60036 mate craneOperator >Job: Job_82_11_72 60026 captain >Job: Job_28_11_75 60024 craneOperator inspector engineer stevedore >Job: Job_39_56_39 60023 >Job: Job_78_36_10 60025 mate clerk	ScreenShots: fig.3.1	Pass
10	skill search: “crew” name search: “job”	displays: Search Results: >Job: Job_66_52_86 60015 crew >Job: Job_93_37_24 60012 engineer crew inspector >Job: Job_80_20_52 60042 crew >Job: Job_87_38_77 60035 crew carpenter	ScreenShots: fig.3.2	Pass
11	index search “60035” skill search: “crew” name search: “job”	displays: Search Results: >Job: Job_87_38_77 60035 crew carpenter	ScreenShots: fig.3.3	Pass

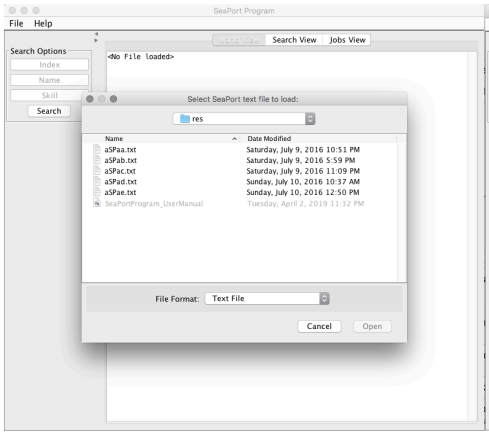
ScreenShots



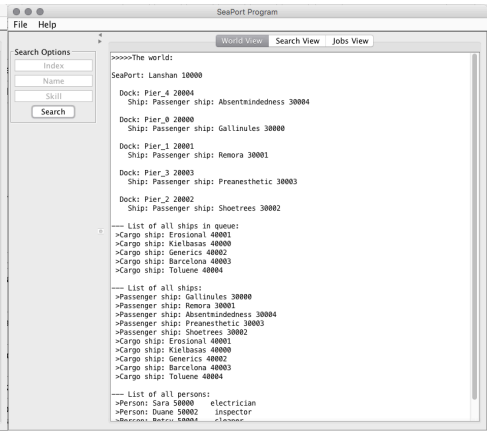
(fig. 1.0)



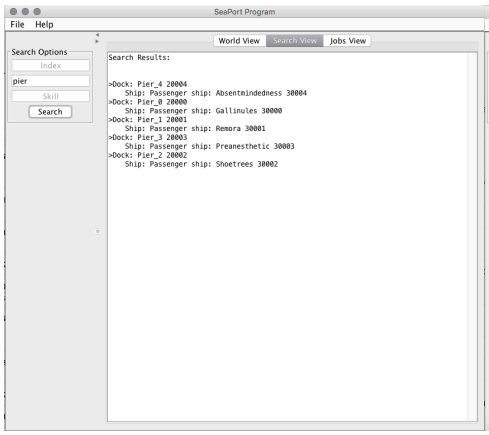
(fig. 1.1)



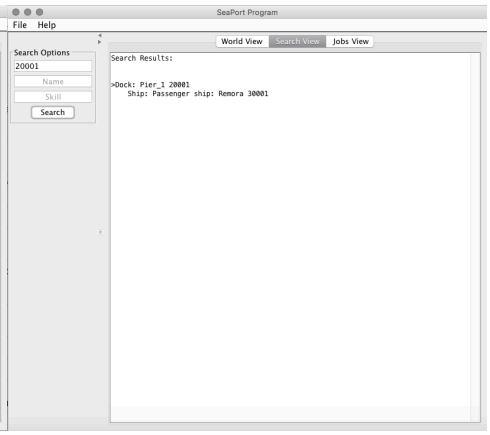
(fig. 1.2)



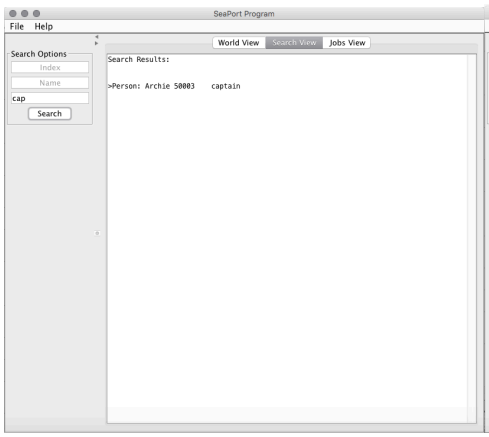
(fig. 2.0)



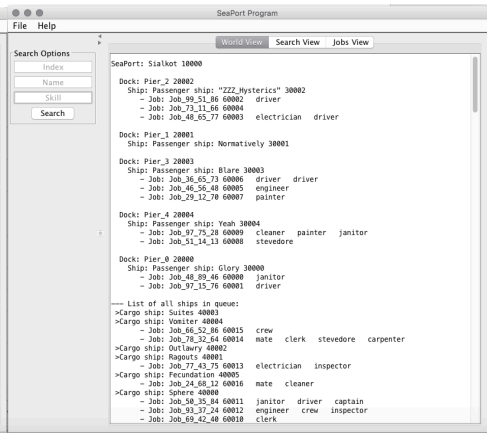
(fig. 2.1)



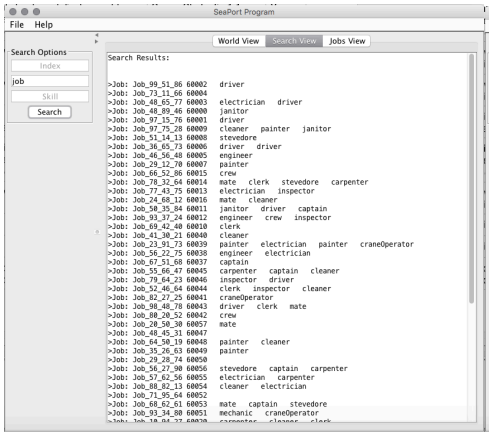
(fig. 2.2)



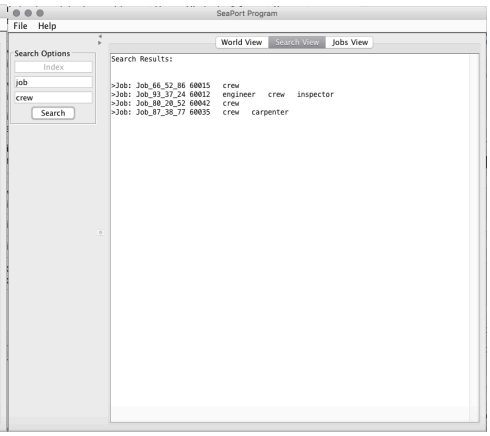
(fig. 2.3)



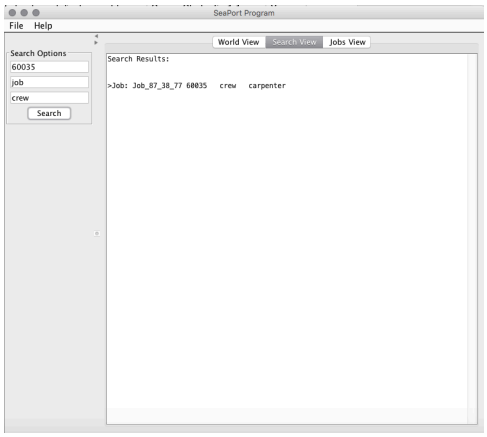
(fig. 3.0)



(fig. 3.1)



(fig. 3.2)



(fig. 3.3)

Reflection and Lessons Learned

This Project was one of the larger coding projects I have done for school. The project also builds on itself as the course progresses; That creates a situation I haven't encountered in other classes, writing code that needs to be maintained and expanded. This provides a good opportunity to focus on code that is easy to read and has strong comments explaining the code.

As a result I shorted methods and tried to design them with only a single purpose to ensure methods are reusable and can be used for future expansion without conflicting with existing code.

The code fragment templates provided great insight in how to code the project. I often saw shorter was of doing things but tried to mimic the suggested code closely as to avoid future problems; However, I also felt like this limited me in many ways. I would prefer to use other data structure in certain places. I often had it in my mind how using a SQL database with my code would be really efficient. Additionally, the required data file format bugged me a bit too. I would much rather prefer an xml file. With an xml I code tag entities in the file and parse them pretty easily with SAX or DOM.

One of the areas I have been trying to improve is Java 8+ I have been trying to expand my use and familiarity with Java 8+ features, such as streams and lambdas. As such, I used stream instead of for-loops in most places.

Overall, this was a fun project to code, albeit somewhat time consuming. I gathered a lot of experience from this project and learned a new way to look at inheritance.