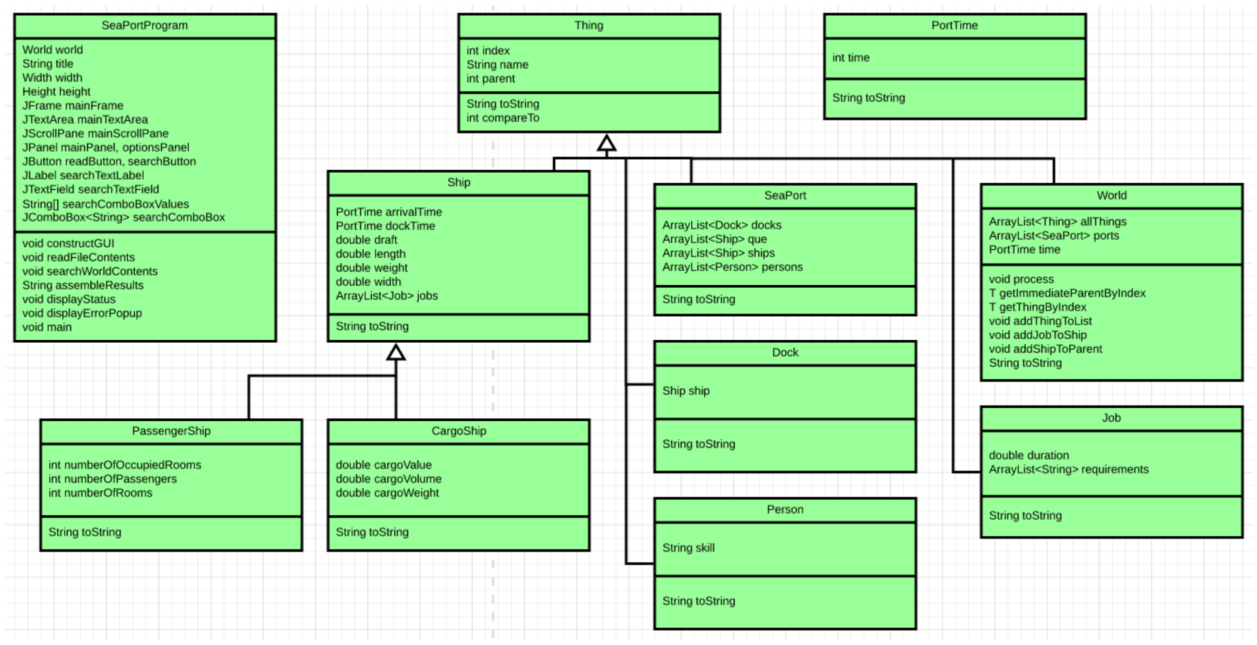
Project 1 Documentation

Andrew Eissen

1/15/2018

UMUC CMSC 335

Part I: Design



(Author note: Getters/setters not included in UML diagram due to a lack of space and an inability to make the entire diagram fit in one frame to screen capture. Relatedly, all images, diagram and screenshots included, have been included in a zip file for convenient HD viewing.)

As per the first page of the project design rubric, the author assembled and included all classes as they appear. The only unused class was `PortTime` which was mentioned nowhere in the rubric and was simply included due its mention on the first page. The only private class field not on the rubric's listing on fields/methods is `allThings`, an `ArrayList` of `Things` contained within new `World` objects and used to store a quick and convenient listing of all `World` extant objects. This was added to simplify the retrieval process of objects during user searches and provide a comprehensive listing of all objects of all types as they exist in the file.

Part II: User's Guide

User unfamiliar with the process of running Java programs may have some difficulty running `SeaPortProgram` Project 1 without some prior experience and a set of qualifications. Users must have Java installed on their computer and must have administrator privileges on their chosen machine. Pointing the Windows command prompt to the folder containing the files, users must first compile the classes by typing `javac SeaPortProgram.java`. Once compiled, users may enter `java SeaPortProgram` to run the program and begin interacting with the GUI. Alternatively, if users possess an IDE like NetBeans, they can open `SeaPortProgram.java` and run the program from within after creating a new package.

To prevent invalid input, the program will not open non-text files, so users must make an effort to open a file that conforms to the organizational layout of the sample data files. It should be noted that the use of search bar necessitates the inclusion of proper input. Case and spelling are both integral to retrieving the desired results, as the program will not find results that are improperly formatted as they appear in the text file.

Part III: Test Plan

Starting out, the author sat down and sought to visualize the underlying processes at work in the text file and the relationships between objects of different types. Laying out the design in a diagram of sorts, the author was more easily able to visualize the connection between types of objects and the relationship each of their classes would possess with others of similar types.

While the process of passing objects to their proper containers during the file reading part of the assignment was straightforward enough, the author had some initial difficulty in deciding how best to search through objects. Though the preferred option was the use of a set of `HashMaps` to aid in searching for a specific object by name or index, the author elected to leave this approach for the following projects and pursue a less optimized but more rubric-friendly approach. Using a number of generic methods aimed at iterating through `ArrayLists` of `Thing` objects in nested loops, the program makes use of the predefined relationship class structure at play as well as a listing of all encountered `Things`.

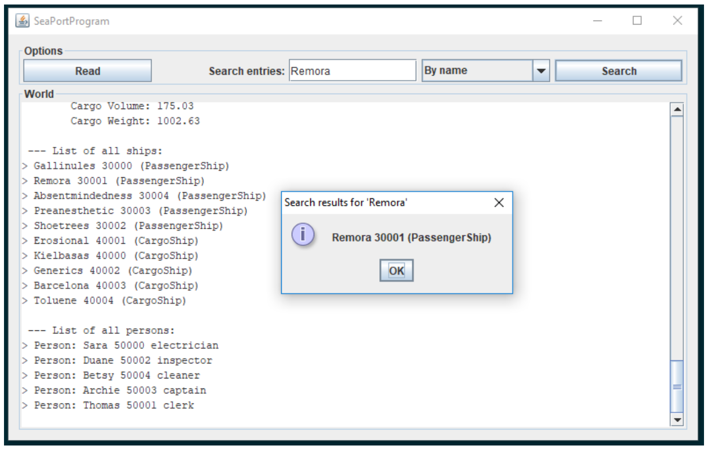
Making use of the rubric's suggested methods, the author initially had several different but near-identical methods aimed at either adding objects to their proper classes or retrieving an object by an index. Though this process worked and produced satisfactory results as expected, the sheer amount of needless code repetition made the author consider the possibility of using generics to accept `Things` of all types. As such, the half-dozen or so near-identical methods were all replaced by three generic methods that retained the same purpose as those in the rubric but were slightly more optimized and cleaner in form.

Part III b: Test Cases

The test cases included herein were retrieved from tests run on two data files, namely `aSPaa.txt` from the projects.zip download in the Contents tab, and from `asPad.txt`, assembled by the author using the `CreateSeaPortDataFile.java` program. The first four tests used the first file, and the last test employed the second file in question.

Input	**Expected Results**	**Results**
Remora, name	Remora 30001 (PassengerShip)	Remora 30001 (PassengerShip)
UMUC, name	Warning: UMUC not found.	Warning: ‘UMUC’ not found.
50004, index	Betsy 50004 (Person)	Betsy 50004 (Person)
electrician, skill	Sara (id #50000)	Sara (id #50000)
<no input>	Error: No world initialized. Please try again.	Error: No world initialized. Please try again.
mate, skill	Mark (id #50005) Joe (id #50023) Gabriel (id #50070) Gayle (id #50007) Mack (id #50055) Brendan (id #50068) Patrick (id #50037)	Mark (id #50005) Joe (id #50023) Gabriel (id #50070) Gayle (id #50007) Mack (id #50055) Brendan (id #50068) Patrick (id #50037)

Test Case 1

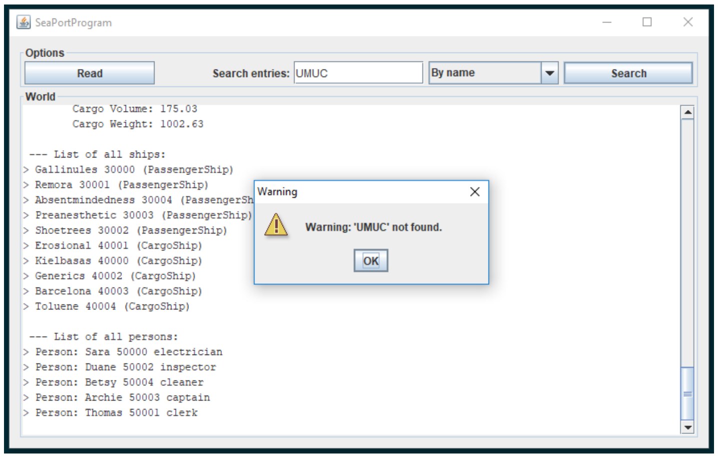


```
        Cargo Volume: 175.03
        Cargo Weight: 1002.63

 --- List of all ships:
> Gallinules 30000 (PassengerShip)
> Remora 30001 (PassengerShip)
> Absentmindedness 30004 (PassengerShip)
> Preanesthetic 30003 (PassengerShip)
> Shoetrees 30002 (PassengerShip)
> Erosional 40001 (CargoShip)
> Kielbasas 40000 (CargoShip)
> Generics 40002 (CargoShip)
> Barcelona 40003 (CargoShip)
> Toluene 40004 (CargoShip)

 --- List of all persons:
> Person: Sara 50000 electrician
> Person: Duane 50002 inspector
> Person: Betsy 50004 cleaner
> Person: Archie 50003 captain
> Person: Thomas 50001 clerk
```

Remora 30001 (PassengerShip)

The first test case concerns the assembly of a legitimate file and proper search input, namely “Remora.” The program displays a popup window with information pertaining to this ship as expected.

Test Case 2

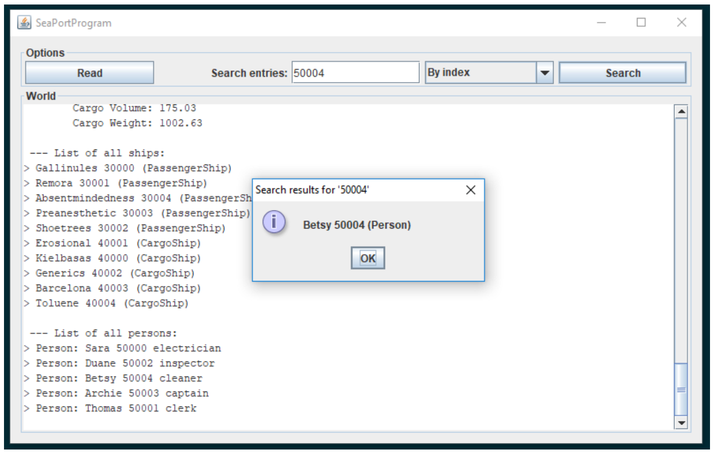


```
        Cargo Volume: 175.03
        Cargo Weight: 1002.63

 --- List of all ships:
> Gallinules 30000 (PassengerShip)
> Remora 30001 (PassengerShip)
> Absentmindedness 30004 (PassengerSh
> Preanesthetic 30003 (PassengerShip)
> Shoetrees 30002 (PassengerShip)
> Erosional 40001 (CargoShip)
> Kielbasas 40000 (CargoShip)
> Generics 40002 (CargoShip)
> Barcelona 40003 (CargoShip)
> Toluene 40004 (CargoShip)

 --- List of all persons:
> Person: Sara 50000 electrician
> Person: Duane 50002 inspector
> Person: Betsy 50004 cleaner
> Person: Archie 50003 captain
> Person: Thomas 50001 clerk
```

Warning: 'UMUC' not found.

The second test handles cases wherein a proper file has been read and assembled, but one where the user's chosen search term does not exist in the database. As a result, the program displays a warning popup to notify the user of this fact.

Test Case 3

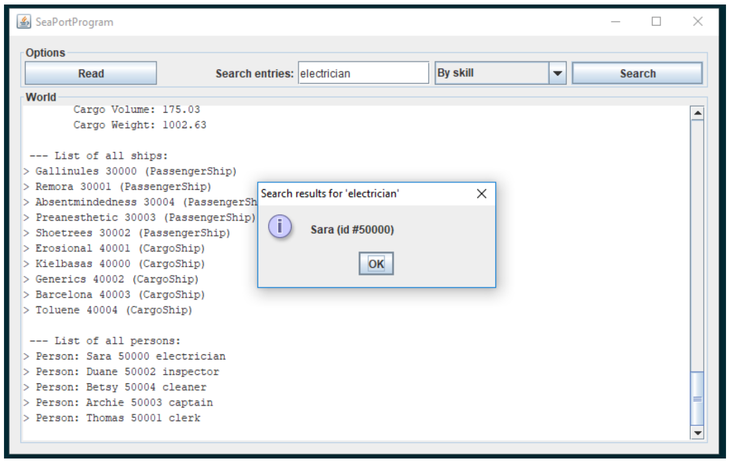


```
        Cargo Volume: 175.03
        Cargo Weight: 1002.63

 --- List of all ships:
> Gallinules 30000 (PassengerShip)
> Remora 30001 (PassengerShip)
> Absentmindedness 30004 (PassengerSh
> Preanesthetic 30003 (PassengerShip)
> Shoetrees 30002 (PassengerShip)
> Erosional 40001 (CargoShip)
> Kielbasas 40000 (CargoShip)
> Generics 40002 (CargoShip)
> Barcelona 40003 (CargoShip)
> Toluene 40004 (CargoShip)

 --- List of all persons:
> Person: Sara 50000 electrician
> Person: Duane 50002 inspector
> Person: Betsy 50004 cleaner
> Person: Archie 50003 captain
> Person: Thomas 50001 clerk
```

Betsy 50004 (Person)

The third case tests the program's response to searches by index rather than by name, as was the case with the previous searches. In this case, the program displays the chosen `Thing` with the selected index as expected.

Test Case 4



electrician

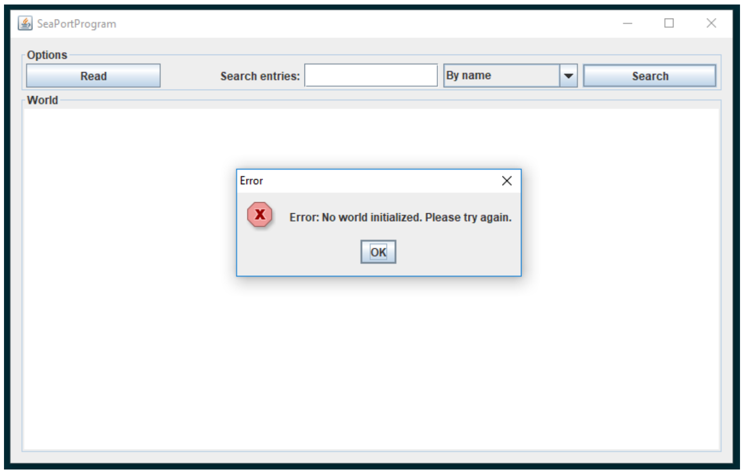
```
        Cargo Volume: 175.03
        Cargo Weight: 1002.63

 --- List of all ships:
> Gallinules 30000 (PassengerShip)
> Remora 30001 (PassengerShip)
> Absentmindedness 30004 (PassengerSh
> Preanesthetic 30003 (PassengerShip)
> Shoetrees 30002 (PassengerShip)
> Erosional 40001 (CargoShip)
> Kielbasas 40000 (CargoShip)
> Generics 40002 (CargoShip)
> Barcelona 40003 (CargoShip)
> Toluene 40004 (CargoShip)

 --- List of all persons:
> Person: Sara 50000 electrician
> Person: Duane 50002 inspector
> Person: Betsy 50004 cleaner
> Person: Archie 50003 captain
> Person: Thomas 50001 clerk
```

Search results for 'electrician'

Sara (id #50000)

This test case handles retrieving all instances of `Persons` with the inputted skill/profession. In the case of `aSPaa.txt`, there is only one `Person` with the skill of electrician, so that user is displayed in the popup modal.

Test Case 5

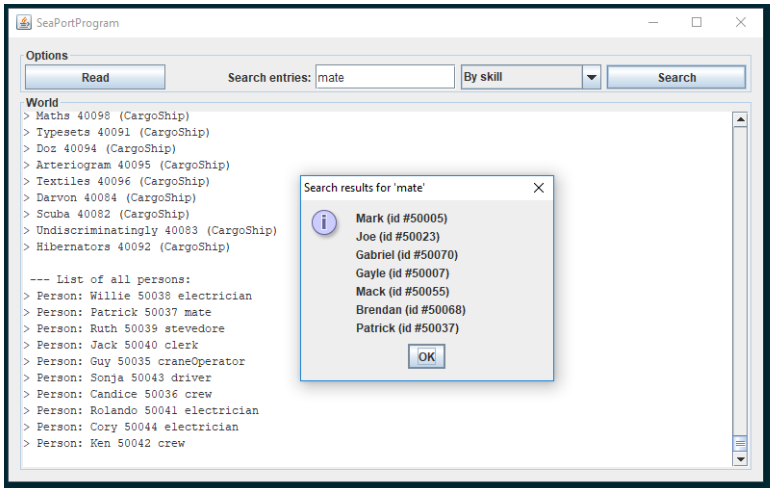


Error

Error: No world initialized. Please try again.

The fifth test case handles cases wherein the user may have accidentally failed to select a file and build a world before pressing the “Search” button. Since there is no data, the program displays an error window notifying the user of this fact.

Test Case 6



Search entries: mate
By skill

World
```
> Maths 40098 (CargoShip)
> Typesets 40091 (CargoShip)
> Doz 40094 (CargoShip)
> Arteriogram 40095 (CargoShip)
> Textiles 40096 (CargoShip)
> Darvon 40084 (CargoShip)
> Scuba 40082 (CargoShip)
> Undiscriminatingly 40083 (CargoShip)
> Hibernators 40092 (CargoShip)

 --- List of all persons:
> Person: Willie 50038 electrician
> Person: Patrick 50037 mate
> Person: Ruth 50039 stevedore
> Person: Jack 50040 clerk
> Person: Guy 50035 craneOperator
> Person: Sonja 50043 driver
> Person: Candice 50036 crew
> Person: Rolando 50041 electrician
> Person: Cory 50044 electrician
> Person: Ken 50042 crew
```

Search results for 'mate'
```
Mark (id #50005)
Joe (id #50023)
Gabriel (id #50070)
Gayle (id #50007)
Mack (id #50055)
Brendan (id #50068)
Patrick (id #50037)
```

As an illustration of the program's ability to display multiple instances of shared input data, the text file `aSPad.txt` was used to display a list of `Person` instances with the skill of mate. As expected the popup window displays each in a list.

Part IV: Comments

As expressed in an email to the professor, the author was of the opinion that the use of a set of related `HashMaps` organizing input by index, name, etc. would have made searching for desired objects that much easier and more efficient than iterating over the entire `World` with nested looping structures and utility methods. From the author's quick glance at the following projects, it appears the author may get a chance to implement this idea in future.

As far as improvements over existing functionality go, the author may wish to revamp and improve upon the structure of the generic methods contained within the body of `World`. As they were integrated late in the game and expanded as the author learned more about generics and reflection, they appear somewhat messy in comparison to some of the starting methods that did not change as the project came together. Some more research into generics and reflection is required by the author to more naturally employ such powerful tools in future.

Part V: Lessons Learned

The author learned a number of things related to the implementation of his design plan during the course of this assignment. Most significant was probably the use of reflection to pass the name of a method to be used by a generic method to properly invoke the desired `ArrayList` getter. The author, experienced in the use of front-end web development languages like JavaScript, has passed method names to functions before (generally when passing a handler to be used as a callback during an AJAX call) and had no idea prior to this assignment that Java had the means of doing this as well. The ability to pass method names to be invoked within a single generic method decreased the amount of copy/pasted code within the body of the `World` class and allowed the author to consolidate such near-identical functions into a single set of methods.

Similarly related to this was the use of generics in the program. Though the author has used generic methods and classes in his programs in the past, he has never really fully grasped the power of generics in Java prior to the assignment. Though he still struggles with some of the aspects of generics and casting (namely visualization of the process actually happening), he was able to successfully reduce the number of near-identical methods through the application of such practices.