CSE 1325-002
Spring 2015
Homework #3

~~Assigned February 24th, 2015, 2:00 pm~~

~~Achievement on March 5th, 2015, 2:00 pm~~

Assigned March 3rd , 2015, 2:00 pm

Due: March 24th, 2015, 11:59 pm

NO DELAYS, NO LATE ASSIGNMENTS, SPARE TIME HAS BEEN USED UP.

1. Introduction

For this assignment, you will be updating your Shipping text program in order to continue your education in Object-Oriented Programming and Java. For this assignment, the ships, docks, cargos, ~~and exceptions~~ will be expanded by using inheritance. You will have to reuse the basic set of classes and menus, and then have similar features. In addition, you will be expected to create a class diagram from your code.

2. Objectives

By the time you complete this assignment, you should be familiar with:

- Inheritance Properties
- Usage of **instanceof**
- Demonstrate, as necessary, Object Oriented Properties: ~~abstract vs concrete, static vs dynamic~~, overloading vs overriding, parent and child, and *polymorphism*
- ~~Custom Exception Handling~~
- Class Diagrams

3. Requirements to turn in assignment

Netbeans has an export project to a zip file ability. When turning in the assignment, please zip the entire project and upload to Blackboard, or other instructions as needed. Be sure to complete the first menu task, for this is where your name is on your paper. The Diagram may be uploaded as a separate file, or include a readme.txt or comments for the grader. Be sure to do a Run->Clean and Build Project *first*. Please see the Net Beans 8.0.2 Export Project Tutorial.

NOTE: If you do not turn in your java files, you will receive a ZERO on the assignment.

3.1. Required Comments:

At the beginning of each file, you should ensure your name and student id number appear at the top of the file.
At the beginning of each class, you should write a short description of the class
At the beginning of each function, you should write a short description, and include the parameters and returns of the function. If there are no parameters or returns, leave that part blank.

3.2. Good Naming Standards

Writers of Java code have established some simple standards for how a program should be written, addressing both the functionality and the readability of the code. You can find several standards online by simply searching for Java coding standards, including ones by Oracle and Google. Additionally many corporations who use Java have established their own, slightly modified, set of coding standards.

1. Variable names will be written in camel case (e.g. mapData or mainMapMenu). This includes both the member variables of your classes and the internal variables used by a method, such as the parameters.
2. Class names will be written with the first letter of each word capitalized (e.g. MainMenu, MapNode).
3. Class and variable names will be in plain language and will not be abbreviated (e.g. instead of calling an instance of MainMenu simply mm you would call it mainMenu).
4. Any member variables of a class will be set as "private" variables. In order to access those variables getter and setter methods will be created.
   e.g.

```
public class MapData
{
     private MapNode mapNode;

     public MapNode getMapNode()
     {
          return mapNode;
     }
     public MapNode setMapNode(MapNode newMapNode)
     {
          this.mapNode = newMapNode;
     }
}
```

4. Instructions

In order to complete this assignment, three key issues must be done.

- First, you must create a Class Diagram of the system
- Second, you must create and update the object-oriented classes.
- Third, you must update a menu system that manipulates these classes.
- Fourth, you must ensure the functions for the menu are correct
- ~~Fifth, you must create a custom exception using inheritance~~

4.1. Part 1. Class Diagram

Every real object-oriented project has a Class Diagram from the Unified Modeling Language (UML). For this homework, you will be creating your first Class Diagram, according to the rules we go over in class.

Rules of Thumb:

Each Class is an entity in the diagram, and has a box of the diagram. This box should have three key areas. The first zone of the box contains the name of the class. The second area of the box is the member variables. The third level of the box includes the active functions. It is perfectly fine to leave the get and set functions off of your class diagram in this assignment. But key functions should be listed. API classes, such as Strings and Arraylists, may be member variables but do not need detailed entries on the class diagram.

The second component of the Class diagram are the relationships: Association, Aggregation, and Inheritance. Association means “This class is used by the other class” In such situations, the class is used in a calculation, function call, or other process of the other class. The association relationship is a black line. Aggregation means “it is a part of”. For example, a brick is part of a wall. This can be determined by thinking if the memory is assigned in the class. If the new command is used to instantiate an object within the current class, that object is part of the current class. The symbol for aggregation is a blue line, with a solid blue diamond at the end of the line with the owning class. The third relationship is inheritance. If one class is the subclass or child of the current class, a red line goes from the current class to the child class. The symbol on this line will be an open red triangle, with the point going towards the parent. See the class presentation for examples. In addition, aggregation and association may have markers of one-to-one (1 and 1), one to many (1 and *) and many-to-many (* and *) relationships.

Description

Your class diagram should match your current design. Every class you have written yourself will be on this diagram. Key classes, from homework 1, homework 2, and homework 3 are:

Homework #1.	Main, Ship, Dock, Cargo
Homework #2.	Port, Map, FileHandler
Homework #3.	Oil Tankers, Container Ships, Piers, Cranes, Boxes, Oil

~~No, you do not need to put your custom exception on the class diagram.~~

4.2.Part 2. Creating and Updating Classes

Use a Java system, ideally NetBeans, to create the following classes for use in your program.

~~First, packages should be created for the Port and its types of Dock, and a package should be created for the Map and its types of ships, and a package should be created for the different types of cargo.~~

The Cargo Class

- The Cargo class will now become a parent class.
- String description should now be of access protected.
- The default description should be “Bananas!”
- Double tonnage should now be of access private.
- The default tonnage should be 10.0
- Should have three different public overloaded constructors:
    - With no parameters
    - String and double parameters
    - String with comma separated values
- Has a public toString() function that has CSV for description and tonnage

```
Bananas!,10.000000
```

- Has a public display() function that shows “X tons of description” to the screen

Example

```
10.00 tons of Bananas!
```

The Box Class

- The Box Class will inherit from the Cargo Class
- The default description should be “Marble”
- Has a private integer values of TEUs (twenty-foot equivalent units)
- The default TEU will be 10000
- Should have three different public overloaded constructors:
    - With no parameters
    - String and double parameters
    - String with comma separated values
- Has a public toString() function that has CSV for description and teu
- Example

```
Marble,10000
```

- Has a public display() function that shows “X TEUs of description” to the screen
- Example

```
10000 teus of Marble
```

The Oil Class

- The Oil Class will inherit from the Cargo Class
- The default description should be “Light Crude”
- Has a private integer values of barrels
- The default barrels will be 700000
- Should have three different public overloaded constructors:
  - With no parameters
  - String and double parameters
  - String with comma separated values
- Has a public toString() function that has CSV for description and barrels
- Example

```
Light Crude,700000
```

-

- Has a public display() function that shows “X barrels of description” to the screen
- Example

```
700000 barrels of Light Crude
```

The Dock Class

- The Dock class will now become a parent class.
- All the previous member variables will become of type protected access
- A new character member variable, a dock symbol, will be a protected
- The dock symbol default will be ‘D’
- Should have three different public overloaded constructors:
  - With no parameters
  - String and double parameters
  - String with comma separated values
- Has a public toString() function that has CSV output.
  Example

```
Langton Dock,N,5,100.000000,15.000000,6.000000,-3.009200,53.443600
```

- Has a public display() function that shows the dock to the screen:

Example

```
First Dock
Dock Number: 100
Size: 100.00X6.00X15.00 metres
Location ( -3.011300, 53.428000)
Location (27,18)
```

The Crane Class

- The Crane class will inherit from the Dock class.
- The member variables will be inherited
- The dock symbol default will be ‘C’
- Should have three different public overloaded constructors:
  - o With no parameters
  - o String and double parameters
  - o String with comma separated values
- Has a public toString() function that has CSV output.
  Example

```
North Crane 3,N,3,100.000000,15.000000,6.000000,-2.999011,53.421044
```

- Has a public display() function that shows the Crane to the screen:

Example

```
North Crane
Crane Number: 100
Size: 100.00X6.00X15.00 metres
Location ( -3.011300, 53.428000)
Location (27,18)
```

The Pier Class

- The Pier class will inherit from the Dock class.
- The member variables will be inherited
- The dock symbol default will be ‘P’
- Should have three different public overloaded constructors:
  - o With no parameters
  - o String and double parameters
  - o String with comma separated values
- Has a public toString() function that has CSV output.
  Example

```
South Pier 2,S,2,100.000000,15.000000,6.000000,-3.013056,53.450606
```

- Has a public display() function that shows the Pier to the screen:
- Example

```
South Pier
Pier Number: 100
Size: 100.00X6.00X15.00 metres
Location ( -3.011300, 53.428000)
Location (27,18)
```

The Cargo Ship Class

- The Cargo Ship class will now become a parent class.
- All of the data members will become protected access
- A new character member variable, a ship symbol, will be a protected
- The dock symbol default will be ‘S’
- Should have three different public overloaded constructors:
    - With no parameters
    - String and double parameters
    - String with comma separated values
- Has a public toString() function that has CSV output.
  Example

```
Cargo Ship,Zenda,Ruritania,0,10.000000,90.000000,10.000000,5.000000,-2.977838,53.410777,Bananas!,10.000000
```

Has a public display() function that shows the dock to the screen

Example

```
Cargo Ship: Zenda
Country of Origin: Ruritania
Transponder: 0
Length: 90.00 metres
Beam: 10.00 metres
Draft: 5.00 metres
Capacity: 10.00 tons
Location ( -2.977838, 53.410777)
Location (65,8)
Cargo:10.00 tons of Bananas!
```

The Container Ship Class

- Has a public toString() function that has CSV output.

Example

```
Container Ship,Zenda,Ruritania,0,90.000000,10.000000,5.000000,-2.977838,53.410777,Marble,10000
```

Has a public display() function that shows the dock to the screen

Example

```
Container Ship: Zenda
Country of Origin: Ruritania
Transponder: 0
Length: 90.00 metres
Beam: 10.00 metres
Draft: 5.00 metres
Number of Holds: 1 tons
Location ( -2.977838, 53.410777)
Location (65,8)
Number of Holds 1
Cargo:10000 teus of Marble
```

The Oil Tanker Class

- Has a public toString() function that has CSV output.
  Example

```
Tanker,Zenda,Ruritania,0,90.000000,10.000000,5.000000,-2.977838,53.410777,Light Crude,700000
```

Has a public display() function that shows the dock to the screen

Example

```
Tanker: Zenda
Country of Origin: Ruritania
Transponder: 0
Length: 90.00 metres
Beam: 10.00 metres
Draft: 5.00 metres
Location ( -2.977838, 53.410777)
Location (65,8)
Barrel Capacity: 700000
Cargo:700000 barrels of Light Crude
```

Exception Handling

~~Exception handling must be implemented for the system from here on out. Please be careful, and put your try and catch blocks in the appropriate places.~~

- ~~When reading from a file, exception handling should be present in case the file cannot be opened, or a file read error. Use a file not found exception to watch the opening of the file. Use a second catch block with a plain Exception to catch other errors that may occur.~~
- ~~When writing to a file, exception handling should be present in case the file cannot be opened, or a file write error. Use a file not found exception to watch the opening of the file. Use a second catch block with a plain Exception to catch other errors that may occur.~~
- ~~For all user input, be aware of type mismatch exceptions. Use a try and catch block with the appropriate TypeMismatch exceptions to handle this error. Use a second catch block with a plain Exception to catch other errors that may occur.~~
- ~~When using a property of a class, surround the entire switch statement with a try/catch block for Null Pointer Exception. Use a "throw" command to pass the exception up to the main menu. There, use a try-catch block with a Null Pointer exception to corral the errors into a single catch block. Use e.printStackTrace() in the main menu catch block.~~

~~In addition, you are to create a custom Exception that will inform the user when they are moving the wrong type of ship at the wrong type of dock. The HarborMasterException will inherit from the java Exception, and contain a message that includes:~~

- ~~The name of the ship~~
- ~~The name of the dock~~
- ~~The longitude and latitude~~
- ~~The Type of the ship~~
- ~~The Type of the Dock~~

4.3 Part 3. The Menu Functions

Now the classes have been defined, it is time to define what the systems should do with these classes.

The Main Menu

```
Main Menu
-------------
1. Show Student ID
2. Load System
3. Ship Menu
4. Port Menu
5. Show Map
6. Display Report

8. Quit
::>
```

Main Menu 1: Show Student ID

Show your name, ID Number, and CSE 1325-002, and the date.
The first item on the menu will be to display your name and student information.

```
Name: Anthony Hope
ID: 1001001894
CSE 1325-002
February 24, 2015
```

Main Menu 2: Load System

The second menu function is Load System. The Load System option should prompt the user to enter a “tag’ that can be sent to the FileHandler class to load in a set of files.

```
Please enter a tag for your load files:
complex
```

The Load System has to bring in two files from the directory.

First, the map file (“complex.map.txt”), as before, will be a set of lines of text that are CSV. The first token will be the column, the second token will be the row, and the third token will be the symbol. A period, ‘.’, to represent open sea, and an asterisk, ‘*’, to represent the land.

Second, the port file (“complex.port.txt”), will have multiple sections. The first line will be 5 CSV fields: port name, number of docks, number of cranes, number of piers, and number of available cargos.

```
Liverpool, 20,5,5,0
```

Each line of the file will then be read, using either a for loop or a system with counters to load each type of file into the right dock type Class before being added to an ArrayList of type Dock in the Port Class.

Main Menu 3: Ship Menu

```

Ship Menu
--------------------
1. Generate Ships
2. Update Ship
3. Display Ships
4. Remove all Ships
5. Previous Menu
====================

```

Ship Menu 1. Generate Ships

Generate Ships should prompt the user to enter a number of ships between 1 and 10. The system should then generate the ships anywhere on the sea in the Map.

```
Please enter the number of cargo ships
15
```

And the properties should be generated by the following conditions.

Location

The longitude and latitude should be selected with a random number generator. The coordinates generated must then be checked against a map. If the ship is at sea, or in a dock, the ship is created. If not, the random generator must be called again until the ship is at sea. Randomly generate ships may have the same longitude and latitude as another ship.

- Longitude: From -3.035000 to -2.988478
- Latitude: From 53.396700 to 53.457561

Identification

The name should be randomly generated using the following table and use **concatenation** to create the ship name. That is, combine a random selection from the First Name column with the random selection of the Second Name column.

Table 1. Ship Names

#	First Name	Second Name
1	Red	Buffalo
2	Green	Pastures
3	Dark	Knight
4	Light	Wave
5	Day	Star
6	Night	Moon
7	Savanah	Lion
8	Mountain	Goat
9	Captain’s	Pride
10	Admiral’s	Joy

Transponder

The Transponder must also be set to a random number.

- Transponder: from 1000000 to 9999999

Ship Class

The type of the ship will be generated at random: Either a Cargo Ship, a Container Ship, or a Tanker. Each generated ship will be put into an ArrayList of Cargo Ship, and use ***polymorphism*** to allow similar vessels to be in the same ArrayList.
Ship sizes may remain at the default values.
Cargo, Boxes, and Oil may remain at the default values.

Ship Menu 2. Update Ship

Update Ship should display a list of all active ships on the map, showing the index in the ship array of the map followed by the ship name. The user can then select the number of the ship, and call the ship properties menu as in homework #1. The selected ship and cargo then can be updated. Students are allowed to reuse their code from homework #1. Include an extra option to position a ship by the row and the column.
The first thing Update Ship must do is to present a ship lookup function, which is to display all the currently available ships.

Example: Ship Lookup Function

```
0. Day Moon
1. Mountain Knight
2. Day Buffalo
3. Captain's Goat
4. Savanah Goat
5. Night Wave
6. Night Lion
7. Captain's Wave
8. Night Knight
9. Dark Joy
10. Mountain Knight
11. Night Wave
12. Savanah Moon
13. Dark Wave
14. Captain's Goat
15. Cancel
===========================
::> 11
```

Ship Properties Menu

```
Ship Properties Menu
---------------------------------
 1. Update Name
 2. Update Registration
 3. Update Transponder
 4. Update Capacity
 5. Update Length
 6. Update Beam
 7. Update Draft
 8. Update Longitude and Latitude
 9. Update Row and Column
10. Update Cargo
11. Display the Ship
12. Previous Menu
=================================
::>
```

Ship Menu 3. Display Ships

Display Ships will show a listing of every ship currently on the map, exactly as in homework #1. In this case, the program should loop through the array of ships, and then display each ship. Students are allowed to reuse their code from homework #1.

```
------------------------------------------------
Container Ship: Day Moon
Country of Origin: Ruritania
Transponder: 5038479
Length: 90.00 metres
Beam: 10.00 metres
Draft: 5.00 metres
Number of Holds: 1 tons
Location ( -3.019200, 53.410611)
Location (18,8)
Number of Holds 1
Cargo:10000 teus of Marble
------------------------------------------------
Tanker: Mountain Knight
Country of Origin: Ruritania
Transponder: 2924499
Length: 90.00 metres
Beam: 10.00 metres
Draft: 5.00 metres
Location ( -3.006911, 53.403656)
Location (32,4)
Barrel Capacity 0
Cargo:700000 barrels of Light Crude
------------------------------------------------
Cargo Ship: Day Buffalo
Country of Origin: Ruritania
Transponder: 3658142
Length: 90.00 metres
Beam: 10.00 metres
Draft: 5.00 metres
Capacity: 10.00 tons
Location ( -3.008667, 53.424522)
Location (30,16)
Cargo:10.00 tons of Bananas!
```

Ship Menu 4. Remove Ships

Have a function that calls the removeAll or clear member function of the ArrayList of ships. This function will clear the map of all the current seafaring traffic.

Main Menu 4: Port Menu

The Port Menu has the following options

```
Port Menu
-----------------------
   1. Update Dock
   2. Unload Ship
   3. Display all Cargos
   4. Display all Docks
   5. Return to previous menu
```

Port Menu 1. Update Dock

Shows a list of all available docks in the port, showing the index of the ArrayList and the name of the dock. The user should then select the index, and be able to update the dock properties as with Homework #1. Students are allowed to reuse code from their first and second homeworks.

```
0. Albert Dock
…
19. Seaforth Dock
20. North Crane 1
…
24. North Crane 5
25. South Pier 1
…
29. South Pier 5
30. Cancel
```

```
Dock Properties Menu
--------------------------------
1. Set the dock name
2. Set the dock section
3. Set the number
4. Set the length
5. Set the width
6. Set the depth
7. Set longitude and latitude
8. Previous Menu
```

Port Menu 2. Unload Ship

The system will check the longitude and latitude of all docks, and of all ships, within the system. This can be done by manipulating the Map class object. If a ship is in a dock, safely, then it will be shown to the user. The user can then select one ship, and the cargo will be moved to the port cargo ArrayList. The cargo for the ship will then be set to **null** afterwards.

```

0. Day Moon
1. Mountain Knight
2. Day Buffalo
3. Cancel
===========================
Please choose a ship

```

Port Menu 3. Display all Cargos

The system will show a number list of all cargos currently in the port cargo array list. The list should show the index of the cargo, and then call the cargo display function as created in homework #1. Students are allowed to reuse the code from Homework#1&2.

```
Cargo at Port
----------------------------------------
0.10.00 tons of Bananas!
1.10000 teus of Marble
2.700000 barrels of Light Crude
```

Port Menu 4. Display all docks

The system will show a number list of all docks currently assigned in the port. The list should show the index of the dock, and then call the dock display function as created in homework #1. Students are allowed to reuse the code from Homework#1&2.

```
Albert Dock
Dock Number: 7S
Size: 100.00X6.00X15.00 metres
Location ( -2.992700, 53.400300)
Location (48,2)

…
Seaforth Dock
Dock Number: 1N
Size: 100.00X6.00X15.00 metres
Location ( -3.025000, 53.458200)
Location (11,35)

North Crane 1
Crane Number: 1N
Size: 100.00X6.00X15.00 metres
Location ( -2.999011, 53.417567)
Location (41,12)

…
North Crane 5
Crane Number: 5N
Size: 100.00X6.00X15.00 metres
Location ( -2.999011, 53.424522)
Location (41,16)

South Pier 1
Pier Number: 1S
Size: 100.00X6.00X15.00 metres
Location ( -3.022711, 53.455822)
Location (14,34)

…

South Pier 5
Pier Number: 5S
Size: 100.00X6.00X15.00 metres
Location ( -3.005156, 53.438433)
Location (34,24)
```

Main Menu 5. Show Map

Each Square on the map is a character that is generated from the file, taken from the object on the background of the map, or found by applying logic.

Symbol	Meaning
D	Unoccupied Dock
S	Cargo Ship safe at sea
C	Unoccupied Crane
B	Container Ship safe at sea
P	Unoccupied Pier
T	Tanker safe at sea
X	Unsafe Ship-See below
$	Ship ready to unload – See Below

```
...................................................D**
..................................................D***
................................................$**D**
...............................................D*D...D
..............................................D***..**
.............................................D****..**
......S.....................................D****..***
.........................................*******..****
.........................................******..*****
.......................X.................*****..******
...............................................*******
..............................................********
.........................................$************
.........................................C************
.............T...........................C************
.........................................C************
.........................................C************
.........................................*************
......................................D***************
......................................****************
..................................D*******************
..................................********************
..................................********************
..................................D*******************
..................................$*******************
.....................................D****************
......B.........................D*********************
.............................D************************
............................P*************************
..........................D***************************
.........................P****************************
.........................P****************************
.........................D****************************
..................D***********************************
..............P***************************************
...........D******************************************
```

The Letter X

A Ships is unsafe and marked with an ‘X’ if:

- Two different ships have the same coordinates
- The ship is at the same coordinates as a dock, and it does not fit in the dock.
- The ship is at a dock of the wrong type.

A ship is ready to unload, and marked with a “$” if the ship fits in the dock, and if the ship is at the right kind of dock.
Hint: You can save some time by converting the longitude and latitude to column and row using the Map Converter.

The Right Type of Dock

Ships much match their docktype

- Docks will be able to unload Cargo Ships
- Cranes will be able to unload Container Ships
- Piers will be able to unload Tankers

There are several possible solutions.
One solution includes using **instanceof** will be able to identify if the ship is at the correct type of dock.

To check the type of dock:

- A pier is an instance of Pier.
- A crane is an instance of Crane
- A dock is not an instance of Pier and is not an instance of Crane

To check the type of ship

- A container ship is an instance of Container Ship
- A tanker is an instance of Tanker
- A cargo ship is not an instance of Container Ship and is not an instance of Tanker

Main Menu 6. Display Report

The display report function will call the display all ship function in the Ship Menu, the display all cargos in the Port Menu, and the display all docks in the Port Menu.

~~Main Menu 7. Save System~~

~~The seventh menu function is Save System. The Save System option should prompt the user to enter a "tag' that can be sent to the FileHandler class to load in a set of files.~~

```
~~Please enter a tag for your save files~~
~~complex2~~
```

~~The system should now save three files: A snap shot file that copies the output of the map display into a text file. (complex2.snapshot.txt). Each line of the snapshot file is one row of the display map.~~

~~The second save file will be a ship file ("complex2.ship.txt") that will show one line of text for each ship.~~

~~The third save file will be a port file ("complex2.port.txt") that will have a first line for the information about the Port name, the number of total docks, and the number of total cargos in the file. This line would then be followed by a listing of all the docks and then all of the cargos.~~

~~A key problem is how to have different types of cargoes and docks when the ArrayList has a polymorphic component. There are three obvious ways to do this. First is to instead of polymorphism, have three arrays for the three ship types, and three arrays for the three dock types, and three arrays for the three cargo types. The second method would be to generate sublists by using the instanceof routine before writing out to the file. The best routine would be to implement the Comparable interface, and then use the abstract compareTo function to sort each ArrayList by Type before writing to file. Any of these systems would work.~~