Homework-7

Objective: Design the Ship Project and create Java classes as follows:

- Ship
- CruiseShip
- CargoShip
- ShipNavigator

Requirements:

- Functionality (85 pts)
 - No Syntax, Major Run-Time, or Major Logic Errors. (85 pts*)
 - *Code that cannot be compiled due to syntax errors is non-functional code and will receive no points for this entire section.
 - *Code that cannot be executed or tested due to major run-time or logic errors is nonfunctional code and will receive no points for this entire section.
 - Clear and Easy-To-Use Interface (5 pts)
 - Users should easily understand what the program does and how to use it.
 - Users should be prompted for input and should be able to enter data easily.
 - Users should be presented with output after major functions, operations, or calculations.
 - All the above must apply for full credit
 - Design a class called *Ship* with the following (20 pts)
 - Instance Variables
 - Name of the ship
 - Launch Date (Format: mm/dd/yyyy) (Hint: read it as a String)

- Methods
 - Accessors and Mutators for all instance variables
 - The mutator for the instance variable *launch date* is only allowed to accept a ship launched during 1990 - 2019 (both inclusive)
 - A toString method that displays the ship's name and its launch date.
 - Next, design another class called *CruiseShip* that is derived from the Ship class (20 pts)
 - Instance Variables
 - Passenger Capacity
 - Number of crew members
 - Methods
 - Accessors and Mutators for all instance variables
 - A toString method that overrides the toString method in the Ship class.
 - It should only display the ship's name, the crew and passenger capacity.
 - Then, create a class called *CargoShip* with the following (20 pts)
 - Instance Variables
 - tonnage (DWT: dead weight tonnage is a measure of how much weight a ship can carry.
 This is different from the ship's own weight)
 - maximum speed
 - Methods
 - Accessors and Mutators for all instance variables

- A toString method that overrides the toString method in the Ship class.
 - It should only display the ship's name, tonnage and maximum speed.
- Finally, create a test class: **ShipNavigator** (20 pts)
 - Contains a main method
 - Creates two objects of the type CruiseShip and one object of type CargoShip
 - Have the user enter the values for the instance variables.
 - Print out the results.
- Coding Style (9 points)
 - Readable Code
 - Meaningful identifiers for data and methods.
 - Proper indentation that clearly identifies statements within the body of a class, a method, a branching statement, a loop statement, etc.
 - All the above must apply for full credit.
- Comments (6 pts)
 - Your name at the beginning of the file as a single-line comment. (1 pt)
 - At least 5 meaningful comments in addition to your name. These must describe the function of the code it is near. (5 pts)

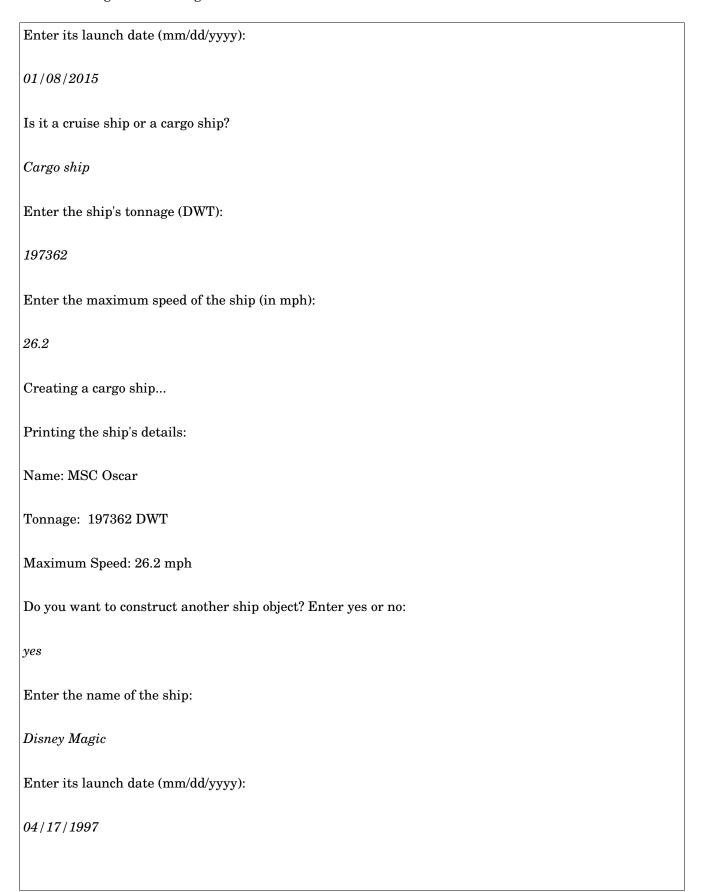
Submission:

• Upload all .java files on Dropbox

Example-1:

Welcome to ShipNavigator!!!





Is it a cruise ship or a cargo ship?
$Cruise\ ship$
Enter the ship's passenger capacity:
2700
How many crew members are on this ship?
945
Creating a cruise ship
Printing the ship's details:
Name: Disney Magic
Passenger Capacity: 2700
Crew: 945
Do you want to construct another ship object? Enter yes or no:
no
Goodbye!
Example-2:
Welcome to ShipNavigator!!!
Enter the name of the ship:
$oxed{Astoria}$
Enter its launch date (mm/dd/yyyy):

09/09/1946
Is it a cruise ship or a cargo ship?
Cruise ship
Enter the ship's passenger capacity:
556
How many crew members are on this ship?
280
Creating a cruise ship
Error: Launch date prior to 1990. Resetting Astoria's launch date to the default 01/01/1990
Printing the ship's details:
Name: Astoria
Passenger Capacity: 556
Crew: 280