Exceptions – Interface – Format output

Homework Assignment # 5

Due next week class 6a at beginning of class

IMPORTANT – You must write one new vehicles and post the *.java code into the discussion group before our next class starts.

Objectives:

This homework will have you gain more experience with inheritance, exceptions and maintaining your previous homework code. There are few additions to this homework. It will be heavily graded on Java docs, UML, and then the code and output. **You will also be making one class that is used by other students in the class, read this section now.** Make sure you have all the requirements completed and correct. If you have any questions, please ask or suffer the consequences.

Maintenance of code:

This homework builds on the existing code you wrote before. Do not rewrite that code - reuse it. Correct any errors indicated in previous homeworks, as they may affect your ability to complete this homework. Maintenance of code consists of fixing and enhancing existing code.

Enhancements *(New)*

Starting with your previous Ordering System classes the enhancements are:

1) Write one new vehicle class of your own choosing and naming. You may create anything from a (non/motorized) unicycle to a NASA crawler using 125.7 US gal/mile (296 l/km) to the Mercedes-Benz Bionic concept vehicle at 84 miles/gallon (2.8 L/100 km). Your class must be something different than Truck, Car, and Boat. It also may not conflict with any other vehicle name by other students in the class. So name your vehicle in the discussion to reserve the names of the one class, then reply to your name entry with the code of the one class.
   Your new classes are to have two choices beyond Model, Color, and Cost, similar to what we have with Truck, Car, and Boat. These options may be for anything to your specific vehicle. You may have fun with these options. You must have at least 2 choices per the two options. Do not give more than two attributes to request, but you may have more than 2 choices per option.
   a. Should the you chose from the other students does not work, it may be a problem with your or their work.
   b. You can chose another class, but you still must report the original class does not work.
2) Your code is to also take one class from a students and include it in your program. If you programmed the menu correct, the addition of new classes should be simple.
3) Write an interface class of vehicle information, **Vinfo**.

   a. Vinfo contains an abstract method **gasMileage()** that returns a double representing the miles per gallon estimation for that vehicle or for a boat; gallons per hour. See **MPG Estimation** section below.
   b. Vinfo contains all the MPG Estimation constants used by the different classes.

4) Formatting of MPG and other numeric output.
5) Java Docs! Check your javadocs to make sure they are complete for every class, and method.
   a. Must have JavaDocs, this means **/** this is a java doc comment */**, not **/* this is not a java doc comment */**
   b. Instance variables and local variables need to be meaningful and where needed commented. Examples: 1) than a variable being "cost", name it "vehicleCost". 2) boolean wfflag=false;   // "wait for flag", allows looping until wwf is true
6) From pervious homework, one menu method is required in Vehicle, `showMenu()`. Have the menu method take care of the **exceptions caught by try/catch** and have the method validate that the user's entry is a valid menu choice. Don't leave the menu code until you have a valid int choice to return
   a. An invalid entry by the user must have a proper error message presented to the user before again asking the user for "Choice:".
   b. In a **try/catch block**, if you are catching the NumberFormatException and the Exception, do not use the same error message.
   c. Catch error in Cost entry of a double number, catch $9,000.00 vs. good 9000.00
   d. Catch menu entry errors, if user types "sedan" rather than a number.

Enhancements (Rehash/reviewed from previous homeworks)

7) No duplication of code in subclass where work can be done in superclass. This means that since Vehicle contains Model, Color, and Cost, the ask/get for that information needs to be performed from the Vehicle class, not the Truck, Car, or Boat. The constructors for these three subclasses automatically call the Vehicle constructor, which is to handle the ask/get for Model, Color, and Cost.
8) All classes have COMPLETE control over their instance variables. This means Vehicle is the ONLY place that uses Model, Color, Cost attributes. The Vehicle "toString()" method is the only place that creates the string of these attributes.
9) All attributes, including those in Vehicle, MUST be private. (Prof. Comment: I've seen homework's where the sub class directly using the super class variables. This is not true object oriented programming. You must use the super's accessors and mutators.)
10) The text that is used in the menu choice arrays (for example: "Sedan", "Coupe", "Wagon") CANNOT appear anywhere else in the entire program or any other class. You need to use the array in the proper methods, such as the menu for displaying the choices, and the getCarType, which uses the array to return the string of the choice.
    a. These arrays must be final static.
11) The result of all menu choices is to store the "int" value as the attribute. Do NOT store the string of the choice. In addition, the text of the menu choices are to all be in the

"private final String [ ]" array. The text strings in the array must NOT appear anywhere else in any code. Reference the array for all usage, including the get methods.

MPG Estimation:

MPG Estimation differs for each type of vehicle. This is a fictitious way to estimate MPG and does not represent any specific vehicle. Since there are different ways to calculate MPG/GPH, the different gasMileage() methods reside in the specific class.

This method is the calculation of gas usage for each vehicle type. There will be a gasMileage() method in each of the four classes.

Vehicle Not meaningful for a generic vehicle. Define as an abstract method.

Boat A boat's MPG for our reference is reported as gallons per hour and is based on boat type.

Pontoon	= 3.5 gallons per hour
PWC	= 2.2 gallons per hour
Sailboat	= 0 gallons per hour

Car MPG is based on car type and towing package:

Sedan	= 23.7	Towing subtract 3 mpg.
Coupe	= 28.2	
Wagon	= 19.5	

Truck This calculation is somewhat complicated and 100% fictional. It is based on the price, $, of the Truck. This is only an excuse to use a Math class. The equation is: `50 – ( SquareRoot( TruckPrice )/10.0 )`

The numbers for gas mileage estimations are subject to change and should be placed in as constants in the class they are used.

`Interface`

Write an interface class, **Vinfo**, which contains an abstract method gasMileage() that returns a double and takes no parameters. This is used by Vehicle for the Truck, Boat, and Car - Gas Mileage calculations.

Vehicle inherits the Vinfo interface. No other classes need to directly inherit Vinfo as other classes will be inheriting the Vehicle class.

`Options for a vehicle (a menu)`

A previous homework required a Menu method in class Vehicle. This HW will set the menu structure requirement to 15 points. If you have a problem, ask the professor for help designing this. Professor will ask "what have you tried/thought about this so far?"

`OOP`

Any access to instance variables (except constants) MUST use accessors or mutators. Even when the variables are accessed within the same class.

Requirements summary:

- All of previous HW requirements, fixed.
- Vinfo interface class, containing:
  - ALL the constants for MPG calculations.
  - The gasMileage() abstract method definition.
- super.toString() used in Car, Truck, and Boat, to have Vehicle supply Model, Color, Cost, and MPG/GPH Estimate. Yes, even though the gasMileage() methods exist in the subclass, we will call them from the superclass.
- All common methods used by Truck, Car, Boat are in Vehicle. This means the Model, Color, and Cost are only handled in the Vehicle class, and Vehicle's toString() method creates the returning string of these attributes as described above.
- Vehicle contains a menu presentation method that accepts (at least) a string constant array of the menu options. At (possibly) most, it would accept the prompt, an error message, and the string constant array of menu options.
- Java Docs – modify and improve where needed, and add Java Docs to the new code.
- Inputs to the program must be using only the Scanner object. The inputs MUST be in the order of the example output and of the correct data type as shown in the sample output (or 15 points deduction). That is, menu's must accept numeric entries for the choices. For example, you may set the Truck Engine Size to any two-size options you want, but they must be asked at the correct time and the number choices must be at least 1 and 2.

- Program names: Orders.java, Truck.java, Car.java, Boat.java, Vinfo.java, your two new classes plus two other java files from the discussion about this assignment, 9 classes! Make sure the menu order is Truck, Car, Boat, followed by your one class, followed by the one class from two different students.

Check your work before submitting:

- One at a time, open your java files in WordPad (not Notepad). You'll find WordPad under Start➔ Programs ➔ Accessories ➔ WordPad. Use TextEdit on a Mac.
- Go to File ➔ Page Setup… and change the printing to Landscape. Print each file to a black and white printer, or view in Print Preview without having to print on paper.
- Review for the following items:
  - Line wrap – a line of text is so long it wraps onto the next printed line.
  - Crammed up code – not enough whitespaces between thought-blocks of code.
  - Indentation – Is what you saw on the screen the same as what is printed? Fix it.
- Java Docs – Create the Javadocs and view them. Make sure the created javadocs have a description for EVERY class and method.

- Javap – run javap against your class name, to see if any variables, or constants appear before the constructor(s). If there are any, make them private variables/constants. For example, for Vehicle.class you would use: C:\> javap Vehicle
- FYI: If you write a Tank class, that starting letter conflicts with Truck. Your Order system should allow for taking that into consideration. Call it some other letter, or use a numbering system menu for this too.

Submitting your work:

- Submit all of your java and class files to the homework DropBox.
- Do not include your JavaDoc files to the DropBox.

Sample Output: (Bolded text was entered by the user)

Your execution will include Truck, Car, Boat, your new class, and one other taken from the homework discussion. Cost error message may be improved of what is shown. This does not show the four extra classes.

```
C:\>java Orders
Your Name Here's Ordering System (218-HW4)

Do you want to order a Truck (T/t), Car (C/c) or Boat (B/b)? c
Entering Car order:
      Model:  Ford
      Color:  Green
      Cost:   $19,875.95
Invalid dollar amount, do not use $ or , in the entered cost. Try again.
      Cost:   19875.95
What type of car is this?
      1. Sedan
      2. Coupe
      3. Wagon
      Choice:     88
Please enter a number 1 through 3
What type of car is this?
      1. Sedan
      2. Coupe
      3. Wagon
      Choice:     3
Does this car have a towing package?
      1. Towing package
      2. No towing package
      Choice:     999
Please enter a number 1 through 2
Does this car have a towing package?
      1. Towing package
      2. No towing package
      Choice:     Fred
Please enter a number.
Please enter a number 1 through 2
      Choice:     2
```

```
Do you want to order another vehicle? y
Do you want to order a Truck (T/t), Car (C/c) or Boat (B/b)? t
Entering Truck order:
      Model:  Dodge RamTruck
      Color:  red
      Cost:   Unknown
Invalid dollar amount, do not use $ or , in the entered cost. Try again.
      Cost:   25000.95
What size truck is this?
      1. Half-ton
      2. Full ton
      Choice:     1
What is the engine size of the truck?
      1. Really big
      2. Not so big
      Choice:     2

Do you want to order another vehicle? Y
Do you want to order a Truck (T/t), Car (C/c) or Boat (B/b)? B
Entering Boat order:
      Model:  WhiteHull
      Color:  White
      Cost:   12500.95
What type of boat is this?
      1. Pontoon
      2. PWC
      3. Sailboat
      Choice:     2
What is the boat's construction?
      1. Wood
      2. Fiberglass
      3. Steel
      Choice:     1

Do you want to order another vehicle? n

Car
      Model:   Ford
      Color:   Green
      Cost:    $19,875.95
      MPG/GPH: 19.5
      Type:    Wagon
      Towing:  No towing package

Truck
      Model:   Dodge RamTruck
      Color:   red
      Cost:    $25,000.95
      MPG/GPH: 34.2
      Load:    Half-ton
      Engine:  Not so big

Boat
      Model:   WhiteHull
      Color:   White
      Cost:    $12,500.95
      MPG/GPH: 2.2
      Type:    PWC
      Made of: Wood

Thank you for using Your Name's Ordering System.

C:\>
```

ISTE-200 Homework 5 – Exceptions – Interface – Format output Grade sheet

Item	Points available	Points earned
Order.java (20%)		
Add *your* new class to menu, successfully entered and printed	10	
Add another student's new class to options, and successfully used (if they work)	10	
Prints the current and new classes summary	5	
Works as expected	5	
Formatted numeric output. All output looks good.	5	
Vinfo.java		
Properly defined class, and getMileage() method.	10	
Vehicle.java		
showMenu() catches number format exception, and other bad data	5	
Your new class		
Written and posted in discussion group	10	
Posted before deadline - next class	10	
Allows entry of common attributes	5	
Class specific attributes are entered properly	5	
Class prints out proper information	5	
Other one class		
Class 1: Works as expected.	5	
Class name: ______________Student name: __________________		
Class 2: Works as expected.	5	
Class name: ______________Student name: __________________		
Boat.java, Truck.java, Car.java		
MPG calculations. Continues to work as expected.	5	
Deductions: Violations of coding standards or poor documentation		
Total	100	