Programming Assignment 6: Creating an Inventory Project
Total Points (50 pts) - Due Monday, November 2nd at 11:59 PM

This programming assignment is intended to demonstrate your knowledge of the following:

- Declare a new class and instantiate an object
- Write instance methods that return a value
- Write instance methods that take arguments

Creating an Inventory Project [50 points]

Create an inventory program that can be used for a range of different products of school supplies (pencils, erasers, markers, notebooks, etc.).

Topic(s):

- Creating classes/objects
- Instance variables/fields
- Constructors
- Methods (getters/accessors, setters/mutators)
- Overloading
- toString() method

You will design an *inventory system* will store these specific products. This table gives you an understanding of the type of data that you will want to store for the attributes of each product.

A. The following table lists 6 example products that you want to store in your system.

Attribute	**Sample Data**
Name of the product (the value that will identify the product in your system).	Pencil Pen Marker Colored Pencil Eraser Notebook
Price (this value holds the price that each item will be sold for).	9.99 8.99 7.75 9.5 4.00 6.75
Quantity of units in stock (this value will store how many of each product item is currently in stock).	25 75 200 500 150 400
Item number (used to uniquely identify the product in your system).	1 2 3 4 5 6

B. The following table gives you an understanding of the type of data that you will want to store for the attributes of each product.

Attribute	Sample Data	Data Type
Name of the product.	Pencil	String
Price.	9.99	Double
Quantity of units in stock.	25	Integer
Item number.	1	Integer

1. Open **Eclipse** and create a project named **inventory**.
2. Create an object class called **Product**.
3. Add the following private instance fields (variables) by using the data types you identified in point **B**:
   a) item number
   b) the name of the product
   c) the quantity of units in stock
   d) the price of each unit
4. Add a comment above the instance field declarations that states:
   ```
   //Instance field declarations
   ```
5. Create two constructors:
   a) A *default constructor* without parameters that will allow the compiler to initialize the fields to their default values. Add a comment above your constructor that explains the purpose of the constructor.
   b) Overload the *default constructor* by creating a constructor with parameters for all four of the class' instance fields so that they can be initialized with values from the driver class. The parameters should be named; number, name, qty, price.
   Example:
   ```
   this.name = name;
   ```
6. Write *getter/accessor* and *setter/mutator* methods for each of the four instance variables. Add comments above them to explain their purpose.
7. Write the *toString()* method to show a description of each **Product** object that includes the instance field values in the following format:
   ```
   Item Number      : 1
   Name             : Pencil
   Quantity in stock : 25
   Price            : 9.99
   ```
8. Create a Java main class called **ProductTester**.
9. Create and initialize six **Product** objects based on the list in point **A**.
   a. Two of the Products should be created using the *default constructor*.
   b. The other four should be created by providing values for the arguments that match the parameters of the constructor.
10. Using the **ProductTester** class, display the details of each product to the console.
11. Save your project.

Here is a sample run:

```
Item Number : 0
Name   : null
Quantity in stock: 0
Price  : 0.0

Item Number : 0
Name   : null
Quantity in stock: 0
Price  : 0.0

Item Number : 3
Name   : Marker
Quantity in stock: 200
Price  : 7.75

Item Number : 4
Name   : Colored Pencil
Quantity in stock: 500
Price  : 9.5

Item Number : 5
Name   : Eraser
Quantity in stock: 150
Price  : 4.0

Item Number : 6
Name   : Notebook
Quantity in stock: 400
Price  : 6.75
```

Submission Instructions

- Execute the program and copy/paste the output that is produced by your program into the bottom of the source code file, making it into a comment. I will run the programs myself to see the output.
- Make sure the run "matches" your source. If the run you submit could not have come from the source you submit, it will be graded as if you did not hand in a run.
- Use the assignment submission link to submit the source code file.
- Submit the following files:
    - Product.java
    - ProductTester.java