Assignment 1

Create a new project in Java that allows a user to manage a store's product stock.

[marks:27, weight 9%]

There are two parts to this assignment:

- The Stock Class models a product's stock for a store.
- This is a tester class that tests and uses the Stock class.

Class: Stock
Data Members: - productID : String - productName : String - qoh : int - rsp : int - sellPrice : double - buyPrice: double
Method Members: + Stock() + Stock(productID : String, productName : String, sellPrice : double) + Stock(productID : String, productName : String, qoh : int, rsp : int, sellPrice : double, buyPrice: double) + getProductId () : String + setProductId (id : String) : void + getProductName () : String + setProductName (name : String) : void + getQoh() : int) + setQoh(qoh : int) : void + getRsp() : int + setRsp(rop : int) : void + getSellPrice() : double + setSellPrice(sPrice : double) : void + getBuyPrice() : double + setBuyPrice(bPrice : double) : void + reStockFee(bPrice : double, bQuantity: int):double + toString() : String

Notes:

- The product ID stored in the productId member is a String that must consists of 7 characters with 3 digits, followed by a underscore, followed by 3 uppercase letters . For example, "123_ABC". All ID values must follow this pattern, otherwise they are invalid.
- The default ID value is "111_AAA"

- The productName is the product's name. The default product name is " Unknown Product".

- "qoh" is short for Quantity On Hand, and represents the quantity of this product currently in stock. For example, if there are 10 products currently in stock, then the QOH is 10. QOH should not be negative. The default value is 0.

- "rsp" is short for Re-Stock Point. The "rsp" represents the amount that should always be in the stock. If QOH is less than "rsp" then store would order products to stock to ensure QOH more than the "rsp".
- The default RSP is 25.

- The sellPrice represents the selling price per unit of this product. Selling cannot be negative, and the default value is 0.

- The buyPrice represents the buying cost of every unit. It would be used to find the total re-stock fee for the product. It should not be negative and default is 0.

- The default constructor should initialize the data members to their default values.

- The multi-param constructors set the values of the appropriate data members to the param values.

- All mutator methods should only assign the param value if it is valid, otherwise an IllegalArgumentException is thrown with a specific, yet concise and informative error message.

- The toString() returns the Stock object as a String in the following format:

  Product ID (Product Name), QOH: x Buying Price: $x.xx

  where "Product ID" is the productID, "Product Name" is the actual name of the product, x is the value in the QOH member, and $x.xx is the value stored in the buyPrice member (formatted with a $ and 2 decimal places).

Input/Output

This program prompts the user to enter the Stock product ID, product name, quantity on hand, re-stock point, selling price, and buying price.

Use the methods of your Stock class to ensure that the user's input is valid. If any input is not valid, repeatedly prompt the user to enter data until that data is valid for main() method.

[Hints: TIP: Construct a default Stock object before you start getting the user's input. Then, for each user input, use a loop to make sure that particular input doesn't cause an exception when you use the appropriate mutator method:

```
// construct default Stock object

REPEAT:
PRINT "Enter [whatever data member you're asking for]:"
GET userInput
// TRY to assign userInput to the data member using the mutator method
// CATCH any exceptions: display an error msg to user
WHILE the userInput is still invalid]
```

Sample Input/Output

Program Prompts are in GREEN BOLD,
User Input is in BLUE ITALIC,
Program Output is in Black BOLD.

[Colors are for your understanding, do not need to implement color]

Example 1:

```
Enter Product ID: 776_ABC
Enter Product Name: Toy Teddy
Qty On Hand: 14
Re-Stock Point: 20
Selling Price: 15.95
Buying Price: 11.95


776_ABC (Toy Teddy), QOH: 14, Buying Price: $11.95
You should order at least 6 products.

Enter # of units to buy: 10
Total Re-Stock Cost : $119.50
```

Example 2:

```
Enter Product ID: 334_CDW
Enter Product Name: Boys Skate, Blue
Qty On Hand: 67
Re-Stock Point: 20
Selling Price: 100.99
Buying Price: 80.00


334_CDW (Boys Skate, Blue), QOH: 67  Buying Price: $80.00

Enter # of units to buy: 5
Total Re-Stock Cost: $400.00
```

Notes:

After the user has entered all 6 input values:

- Display the Stock object as a String.
- If the product needs to be re-stocked, display the following message "You need to order at least ***n*** *products*." where ***n*** is the difference between QOH and RSP.
- Ask the user how many of the product they'd like to buy, then display the total re-stock cost formatted to 2 decimal places. If the user wants to buy less than the recommended quantity then display a message. If the user enters 0 or negative then just display $0 for the cost.

Submission

You must submit :

1. The NetBeans project that contains your assignment, in a ZIP/RAR file.
2. A text (.txt) document containing all of your source code for both the Stock class and the tester class.

The name of the Zip file should be YourLoginName_Assignment1.ZIP

Evaluation

Your submission will be evaluated based on the following criteria:

Criteria

Functionality
Data members, constructors, accessors / mutators and other methods are written as per mentioned requirements and are functioning as specified. Output is in the mentioned format.

Code Efficiency
Program logic is written concisely and is not cluttered with unnecessary tasks or statements; program structures are correct and done in the most efficient way possible. Appropriate variable names used and proper data types used. Method syntax is correct and methods are coded according to specifications. Uses concepts and techniques discussed in class.

Programming Style:
Code uses proper indentation and spacing. Use of comments/documentation. Adheres to programming standards.

Misc.
All other instructions were followed, including submission instructions above. Techniques discussed in class have been used.