You will create a backend for organizing data for a store. This program will be tested using only your objects and their methods / member functions using a driver program provided.

Requirements

Based on the provided UML diagram (see following slides to learn more about UML) of the class Product, create the followings:

- Create a class for Product and include all the fields (data members) as indicated in the UML diagram.
- Implement all the member methods as indicated in the UML diagram.
  - void addShipment(int shipmentQuantity, double shipmentCost)
    - Add shipmentQuantity to inventory and increase totalCost by shipmentCost. Do not replace totalCost, just increase its value.
    - Your program should catch negative shipment quantity and negative shipment costs.
  - double getPrice()
    - This function will calculate the current price based on the average cost per item over time plus a 25% markup.
      \[
      \text{price} = \frac{\text{totalCost}}{(\text{inventory} + \text{numSold})} \times 1.25
      \]
  - void processOrder(int Quantity);
    - If there is not enough inventory, display a message that there is no enough items in inventory. Otherwise, decrease inventory by quantity and increase numSold by quantity.
    - Your program should catch negative quantity entries.
- Make sure you have your name and Bronco ID at the top of your code

  /*     Name: Jane-Joe
  *      Bronco ID: 12345678
  *      Jon Doe helped me with…………
  */

- For the given StoreDriver.java your program should display the following output,

  New Guava shipment arrives!
  Guava costs $0.31
  Guava current inventory is 38 after 12 sold
  New Guava shipment arrives!
  Guava costs $0.30
  Guava current inventory is 98 after 32 sold

  -----------------------

  New Gala Apple shipment arrives!
  Gala Apple costs $0.38
  Gala Apple current inventory is 72 after 28 sold
  New Gala Apple shipment arrives!
  Gala Apple costs $0.32
  Gala Apple current inventory is 267 after 68 sold

Due: November 15, 2017 by 6.00 PM. submit your Product.java file to Blackboard.

Total Points = 100

- Code complies to requirements: 70 points
- Good coding style: 20 points
- Correctness/Robustness: 10 points