METROPOLITAN STATE UNIVERSITY

ICS 141 - 02: Problem solving with programming
Spring 2023

Assignment 3 - Part1: Objects that contain objects

Out: Wednesday, February, 15th, 2023
Due: Wednesday, March, 1st, 2023

Total points: 25

In this assignment, you will implement a Java application, called **RetailStoreApplication**, that can be used in a retail store. The application is to be implemented using three classes, called **RetailItem**, **CashRegister,** and **RetailStoreDriver**. Name the default package ***retail***. The description of these classes is below.

Problem Description

class RetailItem

The `RetailItem` class holds data about an item in the retail store. The class should have the following instance variables:

1. `description`: a `String` variable that holds a brief description of the item.
2. `unitsOnHand`: an `int` variable that holds the number of units currently in inventory.
3. `price`: a `double` variable that holds the item's retail price in dollars.

Implement the following methods in the `RetailItem` class:

1. A constructor that accepts only two arguments to initialize only the `description` and `price` instance variables and it sets the `unitsOnHand` to zero. Note that the order of inputs to the constructor must be in the same order as the field are listed above.
2. Getter (i.e., accessor) methods for each one of the instance variables.
3. **`toString`**: a method that returns a nicely formatted string description of the item where all the item's attributes are displayed in one line separated by tabs. For example:

```
Jacket                12          59.5
```

4. **addUnits**: a method than takes a positive number (of type `int`) and adds that number of items to `unitsOnHand`. The method does not return anything.
5. **getUnits**: a method than takes a number (of type `int`) and subtracts that number from the item's `unitsOnHand`. The method should return `true` or `false` based on whether the input quantity is less than or greater than the current number of units on hand. For example, if the number units on hand is 10 and the input quantity is 5, then the method updates the units on hand to be 5 (or 10-5) and returns `true`. On the other hand, if the current number of units on hand is 10 and the required quantity is 12, then the method does not update the units on hand and returns `false`.
6. **changePrice**: a method that takes an amount (of type `double`) as input and the method then adds the input amount to the item's price. The method does not return anything. Note that the input amount can be either positive or negative to increase or decrease the item's price. Note also that the method should display an error message and do not change the price if changing the price will cause the price to be equal to or less than zero.
7. **equals**: a method to test the equality of two retail items where two items are considered to be equal if they have the same description and the same price.
8. **totalValue**: a method that calculates and returns the total inventory value for the retail item where the total value equals to the number of units on hand multiplied by the unit's price.

Important Requirements

- The output of our program must be nicely formatted.
- Method names and variable names must exactly match (**case sensitive**) the assignment requirements.
- The program should display your name at the end.
- At the top of the Java files, include a comment with your name, a brief description of the program and what it does and the due date.
- **Add appropriate comments to your code.**
- All code blocks must be indented consistently and correctly. Blocks are delimited by opening and closing curly braces. Opening and closing curly braces must be aligned consistently.
- Variable names should convey meaning.
- The program must be written in Java and submitted via D2L.

Test and verify

- Test your Retail class by adding a **RetailStoreDriver** to the **RetailStoreApplication** and instantiating 2 **RetailItem** objects. Print the details of the objects. Invoke all the methods of the RetailItem class on one of the objects.
- If you are at this point, and have successfully completed Part1. Please move on to Part2.