RAD Women
Final Project Requirements

Description

Our Professional Pantry cookware supply store needs to keep better track of our inventory (Stock). We want a system that can track the items we stock, how many are available, and the current price. We do not want to use Products to implement this functionality.

Requirements

- **User Interface**
    - We have a custom User Interface being built by another team. For this project, we are only responsible for the back-end code. We can use the standard Salesforce User Interface to check our trigger utility functionality as we build. In other words, no UI requirements!

- **Data Setup**
    - Stock Item
        - When you installed the RAD Women package it included a custom object called Stock_Item__c, with the following fields:
            - Description__c
            - Item_Name__c
            - Item_Stock_is_Low__c
            - List_Price__c
            - Minimum_Stock_Level__c
            - Stock_on_Hand__c
        - Add some records and be sure to have at least:
            - Several records with data for all fields
            - A few where the stock level is low (Stock_on_Hand__c is less or equal to Minimum_Stock_Level__c) and a few where it is not.

- **Code**

    - **A Trigger** for the Stock Item object. There is already a trigger in your org called **StockItemTrigger.trigger** . You have two options for how to implement the functionality described below.
        1. Put the code directly in the trigger. There is already a section for processing BeforeInsert and BeforeDelete.

2. Put the code in a handler class. The trigger will need to call the methods in that handler class. You may create a class on your own, or use the StockItemHandler.cls that is already set up.
    a. *Hint: Check out the Contact Trigger & Handler for an example of the structure

- **Business Logic** *(in the Trigger or a Utility Class)*

    - 1. Before a Stock Item can be created, check that there is not already a stock item record with a matching name.
        - If there is already a Stock Item with that name, change the name so that it is the name entered, plus the words "Duplicate Item". We have a process in place to check for these elsewhere.
            - **Alternative (Bonus)**: Instead of renaming, add a Trigger Exception error as described here https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_triggers_exceptions.htm

    - 2. Before a Stock Item is deleted, make sure that the stock on hand is at 0. If it is not, Create a case so that someone is alerted.
        - The case should indicate the name of the item that was deleted, the id, and the number of stock that were on hand when it was deleted in the description. The rest of the case can be configured however you think best.
        - You can check out your Week 6 Bulkification homework on the **AccountTrigger** for an example of a similar pattern.

    - 3. We need a method that can be called from elsewhere in our codebase called getLowStockItems
        - This should return a list of all the Stock Items that have a stock on hand count at or below their minimum stock level.
        - It should include the following fields for the Stock Items it returns:
            - ID
            - Item_Name__c
            - Item_Stock_is_Low__c
            - Minimum_Stock_Level__c
            - Stock_on_Hand__c

- ***Testing/Troubleshooting:*** *For the Trigger/Utility functionality, you can check that your code is working by either using the Salesforce UI to create* *and edit Stock Items, or by creating and inserting/updating Items using Execute Anonymous*

    - **General Tips**: *Comment your code well and pay attention to formatting. Readability is important!*

Optional:

- Test Coverage!
    - Create a test class that covers your trigger and/or utility classes. Cover as much or as little as you have time for, this is just to get your feet wet!
    - Check out the ContactHandler_Test class for a sample, and the assigned reading: Testing Best Practices