Please implement a C# class library of the following class diagram. Each method is described below. Each object must be tested, so you should create the test cases and write separate code to test each class. Feel free to refactor the design and to add appropriate constructors, setters, getters, and operations.

|  |
| --- |
|  |

|  |  |
| --- | --- |
| **Item** | |
| Name | The name of the item. e.g. “Pen” |
| Price | The price of the item. e.g. 1.12 |
| This object has no methods. |  |
| **StockedItem** | |
| Name | The name of the stocked item. e.g. “Pen” |
| UnitPrice | The price of one of this stocked item. e.g. 1.12 |
| AmountInStock | The number of this stocked item in stock. e.g. 10 |
| CreateItem() : Item | Creates an **Item** object using name and unitPrice and then returns it. |
| **ShoppingCart** | |
| Items | An array of **Item**s. |
| GetTotalPrice() : Number | Returns the sum of the prices of all items in the shopping cart. |
| SortByItemName() | Sorts items by item name. |
| SortByItemPrice() | Sorts items by item price. |
| GetNumberOfItems() : Number | Returns the number of items in the shopping cart. |
| AddItem(item) | Adds item to the items array. Ensure that the item name is unique. |
| **Stock** | |
| StockedItems | A collection of **StockedItem**s. |
| GetTotalNumberOfItemsInStock() : Number | Returns the total number of stocked items. This is a sum of AmountInStock. |
| GetTotalPriceOfAllItems() : Number | Returns the total price of all items. This is a sum of unitPrice x amountInStock. |
| SortByName() | Sorts stockedItems by name. |
| SortByTotalPrice() | Sorts stockedItems by unitPrice x amountInStock |
| GetNumberOfStockedItems() : Number | Returns the number of unique stockedItems. |
| SellItem(itemName, count) : Item[] | Searches for itemName in the array of stockedItems, creates an array of Items, reduces the amountInStock by count, and then returns the array of Items. If the item is not found or there are not enough items, an exception is thrown. |
| RestockItem(itemName, count) | Searches for itemName in the array of stockedItems and then increases amountInStock by count. If the item is not found, an exception is thrown. |