**SHOPPING BASKET ASSESSMENT**

**Useful Hints (1)**

**Structure of the Program**

The program may be implemented using three Classes:

* **UI class** – Provides the user interaction for the user input etc.
* **OrderItem class -** This class will be used to maintain a single order.

* **Shopping Basket** class – manages the items in the shopping basket. It will contain a list of order items (i.e. List<OrderItem>) as well as other fields and methods as necessary.

The two classes (**ShoppingBasket** and **OrderItem**) will be used to manage items in the shopping basket.

The **UI** class will be the subject of a further ***Useful Hints***

**OrderItem class**

This class will be used to maintain a single order.

This class needs the following **fields**:

* Product name
* LatestPrice
* Quantity
* TotalPrice

This class requires the following **constructors**:

* **Constructor** - taking a product name and latest price. The quantity should default to one.
* **Constructor** - taking a product name, latest price and quantity.

This class also requires the following methods

The following methods should be created

* **getProductName** - to return the product name.
* **getLatestPrice** and **setLatestPrice -**  to return and set the latest price.
* **getQuantity** - to return the quantity.
* **getTotalOrder** - to return the total price of the order items (i.e. latest price \* quantity)
* **AddItems** - to update the quantity by giving as arguments the latest price and the number of items to add. This method should return the updated quantity.
* **AddItems** - to update the quantity by giving as an argument the number of items to add. The method should leave the current price as it was and return the updated quantity.
* **AddItem -** to update the quantity by one and return the updated quantity.
* **RemoveItems** - to update the quantity by giving as arguments the number of items to remove. Quantities should not be able to be lower than 0. This method should return the updated quantity (or zero).
* **RemoveItem** - to update the quantity by removing one item. Quantities should not be able to be lower than 0. This method should return the updated quantity (or zero).

**Shopping Basket class**

This class will manage the items in the shopping basket. It will contain a list of order items (i.e. List<OrderItem>) as well as other fields/properties as necessary.

This class needs the following **fields**:

* orderItems
* numberOfProducts
* basketTotal
* numberOfItems

This class requires the following **constructor**:

* **Constructor** - taking no arguments, creating an empty shopping basket and initialising any fields within the basket

This class also requires the following methods

* **AddProduct -**  taking arguments for the product name (string), latest product value (decimal) and quantity of products (int) purchased.
* **AddProduct** - taking arguments for the product name (string), latest product value. This version of AddProduct will assume that the quantity will be one.
* **RemoveProduct** - taking arguments of product name and quantity which will decrement the quantity. The quantity cannot go below zero, in which case the product will be removed from the shopping basket. If the product name does not exist in the basket then an exception should be thrown.
* **RemoveProduct** - taking an argument of product name which will remove the product from the shopping basket.
* **ClearBasket** - which will remove all items from the shopping basket.
* **getOrderItems** and **setOrderItems**, - which will give access to all the items in the basket.
* get**NumberOfProducts** - which will give the total number of products in the shopping basket.
* **getBasketTotal** - which will return the current total price of the items in the basket.
* **getNumberOfItems** - which will return the total number of items on order in the shopping basket.
* **getCurrentPrice** - which will take a product name as an argument and return the latest price. An exception should be thrown if the item does not exist in the basket.
* **IsProductInBasket -** which will take a product name as argument and return true/false if the product is currently in the basket.
* **SaveBasket** - will take as argument a filename and a receipt for all the items will be produced. The method will return true/false if the save operation has been successful.

The list of order items should be actively maintained within the class. No order items in the list should have a quantity of zero.