Aggregation of Objects

In the last assignment you implemented the Item class. Now you will implement a Shopping cart class to store Item object. You will use these in a simple simulation.

- **Put both classes in one file called A2.py**
- **Name classes, files and methods exactly as described.** Otherwise your code will not be compatible with the simulation code and my test code and you will lose points.

1. Shopping Cart (22 points)

Create the class `ShoppingCart` to represent a shopping cart. A shopping cart will store a list of `Item` objects. The class should have the following attributes

- `itemList: a` list storing item objects

a) (1 pts) Create a constructor to initialize `itemsList` as an empty list.

Add the following methods to your class:

b) (5 pts) `add(name, price, quantity)` – given name, price and quantity,
   - If `itemsList` already has an item with the given name increase the quantity of that item in `itemList` by the quantity passed in. (in this case we are ignoring the price that is passed in).
   - Otherwise, add a new Item object with the given values into `itemList`

c) (5 pts) `remove(name, amt)` given a name and an amount,
   - If `itemsList` has an item with the given name, reduce that item's quantity by the given amount. If the item's quantity goes to 0 or less than 0, remove that item from itemList.

d) (3 pts) `mostExpensive():` return the item object which has the highest price (single item price, not total price). If `itemList` is empty return `none`.

e) (3 pts) `totalPrice():` return the total price of the items in the cart. This should be the **sum** of `price*quantity` over all items in the cart.

f) (3 pts) `__str__()`: return a string containing the string version of each item from `itemList` on a new line. (Hint: call `str()` on each item object).

g) (2 pts) `size():` return the number of items in the cart with unique names.

2. (5 point) Testing ShoppingCart: Add to your main method in A2.py tests for all methods of the ShoppingCart class. Next to each test write your expected results as a comment. **(Test cases without expected results receive 0 credit)**

- **Be sure not to call magic methods directly by name.** Recall that magic methods (those with names surrounded by __) are called behind the scene. For example, if you print a object, its `__str__` method is called behind the scene, if you use `==` to test whether two objects are equal `__eq__` is called behind the scene.

3. **(5 points) Comment** each method you wrote in the ShoppingCart class using #Purpose: #Input: #Output

4. (3 points) Short answer: Answer this as a comment at the bottom of your file Explain how a2interactive.py uses exceptions to prevent users from entering string values for the price or quantity of an item. Start by investigating how this works by entering strings (repeatedly) and observing what happens. Then look at the code in a2interactive.py to see how it does what it does.

Here is a sample run using `a2interactive.py`. The green text is the user's responses and the black is the program's output.

```
>>> main()
Pick an action:
 a - to add an item
 r - to remove item(s)
 d - to display the cart
 q - to quit and purchase
a
Enter name of item to add: milk
Enter the item price: 3
Enter the item quantity: 1

Pick an action:
 a - to add an item
 r - to remove item(s)
 d - to display the cart
 q - to quit and purchase
a
Enter name of item to add: juice
Enter the item price: 2
Enter the item quantity: 1

Pick an action:
 a - to add an item
 r - to remove item(s)
 d - to display the cart
 q - to quit and purchase
a
Enter name of item to add: milk
Enter the item price: 2
Enter the item quantity: 4

Pick an action:
 a - to add an item
 r - to remove item(s)
 d - to display the cart
 q - to quit and purchase
d
Your cart of cost $ 17.0 contains:
milk quantity: 5.0 price: 3.0
juice quantity: 1.0 price: 2.0
The most expensive item in the cart is:
milk quantity: 5.0 price: 3.0

Pick an action:
 a - to add an item
 r - to remove item(s)
 d - to display the cart
 q - to quit and purchase
r
Enter name of item to remove: milk
Enter the item quantity to remove: 2

Pick an action:
 a - to add an item
 r - to remove item(s)
 d - to display the cart
 q - to quit and purchase
d
Your cart of cost $ 11.0 contains:
milk quantity: 3.0 price: 3.0
juice quantity: 1.0 price: 2.0
The most expensive item in the cart is:
milk quantity: 3.0 price: 3.0

Pick an action:
 a - to add an item
 r - to remove item(s)
 d - to display the cart
 q - to quit and purchase
r
Enter name of item to remove: juice
Enter the item quantity to remove: 2

Pick an action:
 a - to add an item
 r - to remove item(s)
 d - to display the cart
 q - to quit and purchase
d
Your cart of cost $ 9.0 contains:
milk quantity: 3.0 price: 3.0
The most expensive item in the cart is:
milk quantity: 3.0 price: 3.0

Pick an action:
 a - to add an item
 r - to remove item(s)
 d - to display the cart
 q - to quit and purchase
q
You are purchasing:
milk quantity: 3.0 price: 3.0
Your total payment is: $ 9.0
   >>>
```