COMP1006 - Assignment #3

(Due: **Monday, February 6th @ 12 noon**)

In this assignment you will simulate a **Shopper** that goes to a grocery store and buys items. The **Shopper** will select **GroceryItems** and place them into an Array representing a shopping cart and then when buying the items, he/she will pack them into **GroceryBags** which will also use Arrays.

(1) The **GroceryItem** Class

Implement a class called **GroceryItem** that represents grocery items that one would pick up when doing your supermarket shopping. The class has 4 **private** instance variables:

- name of type String representing the name of the item (e.g., "Cascade")
- price of type **float** representing the cost of the item (e.g., $4.79)
- weight of type **float** representing the weight of the item in Kilograms (e.g., 2.4 kg)
- perishable of type **boolean.** A value of **true** indicates that the item needs to be refrigerated or frozen and **false** otherwise.

Implement the following instance methods:

- **public** get methods for the attributes.
- A zero-argument constructor
- A constructor that accepts as arguments the item's name, price and weight. Note that the perishable variable is not indicated here. It should be set to **false** by default.
- Make another constructor to accept all 4 parameters, perishable being one of them.
- A **toString()** method that returns a string representation of the item like this:
  "Cascade weighing 2.4kg with price $4.79"

(2) The **GroceryBag** Class

Implement a class called **GroceryBag** that represents a bag which will hold **GroceryItem** objects. The class has the following **private static** constants:

- MAX_WEIGHT of type double which indicates the maximum weight that the bag can hold (set it to 5 Kg). Make use of this value properly in the code below.
- MAX_ITEMS of type integer that indicates the maximum number of items that can be placed in the bag (set it to 25). Make use of this value properly in the code below.

The class has the following **private** instance variables:

- items which is an array that will hold **GroceryItem** objects.
- numItems which is the number of items in the bag.
- weight of type **float** which returns the total weight of all items currently in the bag.

Create the following instance methods:

- Appropriate **public** get methods
- A zero-argument constructor.
- A **toString()** method that shows the number of items in the bag and the total weight of the bag or outputs *"An empty grocery bag ".*
- A method called **addItem()** which adds a given **GroceryItem** to the bag only if the total weight of the bag is not exceeded by this item.
- A method called **removeItem(GroceryItem item)** which removes the given item from the bag.
- A **heaviestItem()** method that returns the heaviest item in the cart. Return **null** if bag is empty.
- A **has(GroceryItem item)** method that returns a **boolean** indicating whether or not the given item is currently in the bag.

Test your code with the following program and ensure that the output is correct:

```
public class GroceryBagTestProgram {
    public static void main(String args[]) {
        GroceryItem  g1, g2, g3, g4, g5, g6;
        GroceryBag   b1, b2;

        g1 = new GroceryItem("Jumbo Cherries", 6.59f, 1.0f);
        g2 = new GroceryItem("Smart-Ones Frozen Entrees", 1.99f, 0.311f, true);
        g3 = new GroceryItem("SnackPack Pudding", 0.99f, 0.396f);
        g4 = new GroceryItem("Nabob Coffee", 3.99f, 0.326f);
        g5 = new GroceryItem("Fresh Salmon", 4.99f, 0.413f, true);
        g6 = new GroceryItem("Coca-Cola 12-pack", 3.49f, 5);

        b1 = new GroceryBag();
        b2 = new GroceryBag();

        b1.addItem(g1);
        b1.addItem(g2);
        b1.addItem(g3);
        b1.addItem(g4);
        b1.addItem(g5);
        b1.addItem(g5);
        b1.addItem(g5);

        System.out.println("BAG 1: " + b1);
        for(int i=0; i<b1.getNumItems(); i++)
            System.out.println("    " + b1.getItems()[i]);
        System.out.println("BAG 2: " + b2);
        System.out.println("\nHeaviest item in BAG 2: " + b2.heaviestItem());

        b1.addItem(g6);
        b2.addItem(g6);

        System.out.println("\nBAG 1: " + b1);
        System.out.println("BAG 2: " + b2);

        System.out.println("\nBAG 1 contains Nabob Coffee: " + b1.has(g4));
        System.out.println("BAG 1 contains a case of coke: " + b1.has(g6));
        System.out.println("BAG 2 contains a case of coke: " + b2.has(g6));
        System.out.println("\nHeaviest item in BAG 1: " + b1.heaviestItem());
        System.out.println("Heaviest item in BAG 2: " + b2.heaviestItem());
    }
}
```

The output should be as follows:

```
BAG 1: A 3.2720003kg grocery bag with 7 items
   Jumbo Cherries weighing 1.0kg with price $6.59
   Smart-Ones Frozen Entrees weighing 0.311kg with price $1.99
   SnackPack Pudding weighing 0.396kg with price $0.99
```

```
   Nabob Coffee weighing 0.326kg with price $3.99
   Fresh Salmon weighing 0.413kg with price $4.99
   Fresh Salmon weighing 0.413kg with price $4.99
   Fresh Salmon weighing 0.413kg with price $4.99
BAG 2: An empty grocery bag

Heaviest item in BAG 2: null

BAG 1: A 3.2720003kg grocery bag with 7 items
BAG 2: A 5.0kg grocery bag with 1 items

BAG 1 contains Nabob Coffee: true
BAG 1 contains a case of coke: false
BAG 2 contains a case of coke: true

Heaviest item in BAG 1: Jumbo Cherries weighing 1.0kg with price $6.59
Heaviest item in BAG 2: Coca-Cola 12-pack weighing 5.0kg with price $3.49
```

---

(3) The **Shopper** Class

Implement a class called **Shopper** which represents a person that buys grocery items. The **Shopper** should maintain an array (called cart) that contains all the loose **GroceryItem** objects that the shopper is planning to purchase. You should create (and make use of) a **MAX_CART_ITEMS** static constant to represent the maximum number of items that can be in the cart at any time (set to 100).

Create the following methods:

- **public** get methods for the instance variables.
- A zero-argument constructor.
- A **toString()** method that returns a string representation of the shopper in the form: *"Shopper with shopping cart containing 10 items".*

- **addItem()**: adds a given **GroceryItem** to the shopping cart.
- **removeItem(GroceryItem item)**: removes the given item from the shopping cart. Make sure that the array does not have gaps in it (i.e., move the last item in the cart into the removed item's position). (Look at my COMP1405 Java Notes 5 pages 138-141 to avoid messing up the array while removing from it). **http://people.scs.carleton.ca/~lanthier/teaching/COMP1405/Notes/COMP1405_Ch5_ArraysAndSearching.pdf**

- A **packBags()** method which returns an array of packed **GroceryBag** objects and possibly some unpacked **GroceryItem** objects (e.g., a case of coke). The method should use the following packing algorithm: Fill as many bags as are necessary one at a time, each to just below its weight limit. That is take items in order from the cart and place them into an initially empty bag and stop when the next item will cause the bag to exceed its weight limit. Then make a new bag and pack that one until it is full. The items are taken sequentially from the cart, although ... as each item is packed it should be removed from the cart (use the **removeItem()** method). Therefore, the use of the **removeItem()** method will affect the order of the items as they are packed. If an item is too heavy (i.e., exceeds the MAX_WEIGHT) to be placed in a bag (e.g., a case of coke), then it should be left in the cart. In theory, all items in the cart may be too heavy, which means no bags may be packed. Or, all items in the cart are exactly the bag limit in weight, which means the number of bags will equal the number of items initially in the cart. Either way, your method should return an array with a size equal to the number of packed bags.

Make use of the following test program:

```
public class ShopperTestProgram {
    public static void main(String args[]) {
        GroceryItem g1, g2, g3, g4, g5, g6, g7, g8, g9, g10, g11;

        g1 = new GroceryItem("Smart-Ones Frozen Entrees",1.99f,0.311f, true);
        g2 = new GroceryItem("SnackPack Pudding",0.99f,0.396f);
        g3 = new GroceryItem("Breyers Chocolate Icecream",2.99f,2.27f, true);
        g4 = new GroceryItem("Nabob Coffee",3.99f,0.326f);
        g5 = new GroceryItem("Gold Seal Salmon",1.99f,0.213f);
        g6 = new GroceryItem("Ocean Spray Cranberry Cocktail",2.99f,2.26f);
        g7 = new GroceryItem("Heinz Beans Original",0.79f,0.477f);
        g8 = new GroceryItem("Lean Ground Beef",4.94f,0.75f, true);
        g9 = new GroceryItem("5-Alive Frozen Juice",0.75f,0.426f, true);
        g10 = new GroceryItem("Coca-Cola 12-pack",3.49f,5.112f);
        g11 = new GroceryItem("Toilet Paper - 48 pack",40.96f,10.89f);

        // Make a new customer and add some items to his/her shopping cart
        Shopper c = new Shopper();
        c.addItem(g1); c.addItem(g2); c.addItem(g3); c.addItem(g4);
        c.addItem(g5); c.addItem(g6); c.addItem(g7); c.addItem(g8);
        c.addItem(g9); c.addItem(g10);  c.addItem(g1); c.addItem(g6);
        c.addItem(g2); c.addItem(g2); c.addItem(g3); c.addItem(g3);
        c.addItem(g3); c.addItem(g3); c.addItem(g3); c.addItem(g10);
        c.addItem(g11); c.addItem(g9); c.addItem(g5); c.addItem(g6);
        c.addItem(g7); c.addItem(g8); c.addItem(g8); c.addItem(g8);
        c.addItem(g5);

        System.out.println("\nINITIAL CART CONTENTS:");
        for (int i=0; i<c.getNumItems(); i++) {
            System.out.println("   " + c.getCart()[i]);
        }

        // Pack the bags and show the contents
        GroceryBag[] packedBags = c.packBags();
        for (int i=0; i<packedBags.length; i++) {
            System.out.println("\nBAG " + (i+1) + " (Total Weight = " +
                                packedBags[i].getWeight() + "kg) CONTENTS:");
            for (int j=0; j<packedBags[i].getNumItems(); j++) {
                System.out.println("   " + packedBags[i].getItems()[j]);
            }
        }
        System.out.println("\nREMAINING CART CONTENTS:");
        for (int i=0; i<c.getNumItems(); i++) {
            System.out.println("   " + c.getCart()[i]);
        }
    }
}
```

Here is the expected result (although the order of the items being packed may vary according to your algorithm).

```
INITIAL CART CONTENTS:
   Smart-Ones Frozen Entrees weighing 0.311kg with price $1.99
   SnackPack Pudding weighing 0.396kg with price $0.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Nabob Coffee weighing 0.326kg with price $3.99
   Gold Seal Salmon weighing 0.213kg with price $1.99
   Ocean Spray Cranberry Cocktail weighing 2.26kg with price $2.99
   Heinz Beans Original weighing 0.477kg with price $0.79
   Lean Ground Beef weighing 0.75kg with price $4.94
   5-Alive Frozen Juice weighing 0.426kg with price $0.75
   Coca-Cola 12-pack weighing 5.112kg with price $3.49
   Smart-Ones Frozen Entrees weighing 0.311kg with price $1.99
   Ocean Spray Cranberry Cocktail weighing 2.26kg with price $2.99
   SnackPack Pudding weighing 0.396kg with price $0.99
   SnackPack Pudding weighing 0.396kg with price $0.99
```

```
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Coca-Cola 12-pack weighing 5.112kg with price $3.49
   Toilet Paper - 48 pack weighing 10.89kg with price $40.96
   5-Alive Frozen Juice weighing 0.426kg with price $0.75
   Gold Seal Salmon weighing 0.213kg with price $1.99
   Ocean Spray Cranberry Cocktail weighing 2.26kg with price $2.99
   Heinz Beans Original weighing 0.477kg with price $0.79
   Lean Ground Beef weighing 0.75kg with price $4.94
   Lean Ground Beef weighing 0.75kg with price $4.94
   Lean Ground Beef weighing 0.75kg with price $4.94
   Gold Seal Salmon weighing 0.213kg with price $1.99

BAG 1 (Total Weight = 3.251kg) CONTENTS:
   Smart-Ones Frozen Entrees weighing 0.311kg with price $1.99
   Gold Seal Salmon weighing 0.213kg with price $1.99
   Lean Ground Beef weighing 0.75kg with price $4.94
   Lean Ground Beef weighing 0.75kg with price $4.94
   Lean Ground Beef weighing 0.75kg with price $4.94
   Heinz Beans Original weighing 0.477kg with price $0.79

BAG 2 (Total Weight = 3.295kg) CONTENTS:
   Ocean Spray Cranberry Cocktail weighing 2.26kg with price $2.99
   Gold Seal Salmon weighing 0.213kg with price $1.99
   5-Alive Frozen Juice weighing 0.426kg with price $0.75
   SnackPack Pudding weighing 0.396kg with price $0.99

BAG 3 (Total Weight = 4.54kg) CONTENTS:
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99

BAG 4 (Total Weight = 4.54kg) CONTENTS:
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99

BAG 5 (Total Weight = 4.936kg) CONTENTS:
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   SnackPack Pudding weighing 0.396kg with price $0.99

BAG 6 (Total Weight = 4.946kg) CONTENTS:
   SnackPack Pudding weighing 0.396kg with price $0.99
   Ocean Spray Cranberry Cocktail weighing 2.26kg with price $2.99
   Smart-Ones Frozen Entrees weighing 0.311kg with price $1.99
   Nabob Coffee weighing 0.326kg with price $3.99
   5-Alive Frozen Juice weighing 0.426kg with price $0.75
   Lean Ground Beef weighing 0.75kg with price $4.94
   Heinz Beans Original weighing 0.477kg with price $0.79

BAG 7 (Total Weight = 2.473kg) CONTENTS:
   Ocean Spray Cranberry Cocktail weighing 2.26kg with price $2.99
   Gold Seal Salmon weighing 0.213kg with price $1.99

REMAINING CART CONTENTS:
   Toilet Paper - 48 pack weighing 10.89kg with price $40.96
   Coca-Cola 12-pack weighing 5.112kg with price $3.49
   Coca-Cola 12-pack weighing 5.112kg with price $3.49
```

Finally, write an **unpackPerishables()** method in the **GroceryBag** class that removes all the perishable items from the bag. Note that the **GroceryBag** should be modified (changed) by this method in that it will likely have less in it afterwards. The method should not print anything out (that belongs in the testing code) but instead should return an array of perishable **GroceryItem** objects.

Add the following to the **ShopperTestProgram** so that it tests your method:

```
System.out.println("\nUNPACKING PERISHABLES:");
for (int i=0; i<packedBags.length; i++) {
    GroceryItem[] perishables = packedBags[i].unpackPerishables();
    for (int j=0; j<perishables.length; j++) {
        System.out.println("   " + perishables[j]);
    }
}
System.out.println("\nREMAINING BAG CONTENTS:");
for (int i=0; i<packedBags.length; i++) {
    System.out.println("\nBAG " + (i+1) + " (Total Weight = " +
                        packedBags[i].getWeight() + "kg) CONTENTS:");
    for (int j=0; j<packedBags[i].getNumItems(); j++) {
        System.out.println("   " + packedBags[i].getItems()[j]);
    }
}
```

Here is the result which should appear at the end of your output (although this may vary according to how you packed the bags earlier):

```
UNPACKING PERISHABLES:
   Smart-Ones Frozen Entrees weighing 0.311kg with price $1.99
   Lean Ground Beef weighing 0.75kg with price $4.94
   Lean Ground Beef weighing 0.75kg with price $4.94
   Lean Ground Beef weighing 0.75kg with price $4.94
   5-Alive Frozen Juice weighing 0.426kg with price $0.75
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Breyers Chocolate Icecream weighing 2.27kg with price $2.99
   Smart-Ones Frozen Entrees weighing 0.311kg with price $1.99
   5-Alive Frozen Juice weighing 0.426kg with price $0.75
   Lean Ground Beef weighing 0.75kg with price $4.94

REMAINING BAG CONTENTS:

BAG 1 (Total Weight = 0.69000006kg) CONTENTS:
   Heinz Beans Original weighing 0.477kg with price $0.79
   Gold Seal Salmon weighing 0.213kg with price $1.99

BAG 2 (Total Weight = 2.869kg) CONTENTS:
   Ocean Spray Cranberry Cocktail weighing 2.26kg with price $2.99
   Gold Seal Salmon weighing 0.213kg with price $1.99
   SnackPack Pudding weighing 0.396kg with price $0.99

BAG 3 (Total Weight = 0.0kg) CONTENTS:

BAG 4 (Total Weight = 0.0kg) CONTENTS:

BAG 5 (Total Weight = 0.3959999kg) CONTENTS:
   SnackPack Pudding weighing 0.396kg with price $0.99

BAG 6 (Total Weight = 3.459kg) CONTENTS:
   SnackPack Pudding weighing 0.396kg with price $0.99
   Ocean Spray Cranberry Cocktail weighing 2.26kg with price $2.99
   Heinz Beans Original weighing 0.477kg with price $0.79
   Nabob Coffee weighing 0.326kg with price $3.99

BAG 7 (Total Weight = 2.473kg) CONTENTS:
   Ocean Spray Cranberry Cocktail weighing 2.26kg with price $2.99
   Gold Seal Salmon weighing 0.213kg with price $1.99
```

IMPORTANT SUBMISSION INSTRUCTIONS:

Submit your ZIPPED **IntelliJ project file** as you did during the first tutorial for assignment 0.

- YOU WILL LOSE MARKS IF YOU ATTEMPT TO USE ANY OTHER COMPRESSION FORMATS SUCH AS **.RAR**, **.ARC, .TGZ**, **.JAR**, **.PKG**, **.PZIP**.

- If your internet connection at home is down or does not work, we will not accept this as a reason for handing in an assignment late ... so make sure to submit the assignment WELL BEFORE it is due !

- You WILL lose marks on this assignment if any of your files are missing. So, make sure that you hand in the correct files and version of your assignment. You will also lose marks if your code is not written neatly with proper indentation. See examples in the notes for proper style.
