ESS 201 Programming II
Lab 4
Term 1, 2018-19

An online grocery store sells a variety of items and delivers ordered items to the customer's home. While there are many ways of categorizing the items, the store is mainly concerned about whether an item is *perishable* or *non-perishable*, since this affects the way they store, pack and ship the items. For instance, perishable items need to be maintained at the recommended storage temperature.

A customer adds Items to a Cart, and then on check-out, the cart is processed by the Biller. Since the store also charges for shipping, the Biller requests the Packer to provide a cost estimate for shipping, which is then added to the final bill.

The biller does not want to worry about the details of the items: final price of each item only depends on the price per unit and the number of units ordered. Hence, she is not interested in whether an item is perishable or not, but wants to look at all ordered items uniformly. Moreover, if the kinds of items or the number of categories were to change, we would still want to retain the biller code "as-is"

The store charges different shipping rates for perishable and non-perishable items since they are packed differently. Perishable items are charged based on the total weight (in kgs) of all such items. Non-perishables are charged based on the total volume of all the items (for simplicity, we will assume this is the sum of the individual volumes of each item). The total shipping cost is added to the final bill by the biller.

Model this as a set of Java classes, with at least the following:
- Store, that manages Carts, and has Billers and Packers.
- Item - the items sold in the store - which have a name and price-per-unit
- Perishables and NonPerishables, that can appropriately provide weight or volume as appropriate, as well as other properties specific to their types, in addition to all the information available for Items
- Cart: provides separate methods to add Perishables and Non-perishables, specifying the number of units of each item being ordered. Can be queried for list of ordered Items, or specifically for Perishable and non-Perishable items

The main reads in data to mimic the actions of a customer: adds various items and specifies the quantity of each item being ordered. Perishables are sold by weight (kgs) and NonPerishables by number of units.

When the user requests checkout, the store calls the biller to process the cart, who computes the total cost of all the items in the cart, and adds the shipping estimate given by the Packer, and prints out a detailed bill.

The shipping cost is computed by the Packer. Today's deal provides free shipping upto 5 kgs of perishables and Rs. 5 per kg beyond that. Shipping for non-perishables is at Rs. 2 per liter beyond the first 4 liters. (This can change periodically and is typically known only to the Packer dept).

Below are the input formats as well as sample input and output. You can assume that vegetables, fruits and dairy products are perishables, and all other items are categorized as non-perishables.

The input first lists the set of Items available at the store, and contains the name and unit-price of each item, and for non-perishables, additionally, the volume (in liters) of each unit. An "End" indicates the end of the list of items.
This is followed by a series of lines that represent the actions of a single user, and each line lists the name of an item, and the number of units (or weight) of items bought. A "Checkout" indicates that shopping is completed and checkout can be done.

Sample Input

```
Banana        35
Carrots       60
Potato        70
Grapes       120
Butter       210
Detergent 200 2.5
Soap          35 0.4
Notebook      40 0.2
End
Grapes        0.5
Potato        3.0
Detergent 2
Soap          5
Butter        0.5
Carrots       1.5
Checkout
```

Output

```
Grapes        0.5       60
Potato        3.0      210
Detergent     2        400
Soap          5        140
Butter        0.5      105
Carrots       1.5       90
Shipping                8.50
Total                1013.50
```