Assignment 3
Shopping Cart: OOP, Inheritance, and ArrayList
EE422C - The University of Texas at Austin

You will be working in 2 person teams.
Points: 20 points total (analysis-1, design – 6, program -13). Each member of the team gets the same initial score, which is then adjusted by each member's participation level.
Deliverables – Turn in one item per team submitted to CANVAS

1. **Analysis** report due at the end of day (on paper) on the designated due date
2. **Designs** due at the end of day (on paper) on the designated date
3. **Implementation** – final program is due at the end of day (.zip file with all parts contained) on the designated date

Objective: To cement your understanding of classes, objects, inheritance and polymorphism; and to gain more experience with rigorous OOD and working in teams. Also to get some experience with using ArrayList.

Problem Statement: You are required to design and implement a "shopping-cart" mechanism for an online vendor that sells groceries, electronics and clothing. The user will add his/her purchases to this shopping cart. Your program will then compute and report the total price for the shopping cart from the base price, tax (as applicable) and shipping cost for each item.

Model Details:

Every purchase item (base-class) bought has a name, price, quantity and weight attributes associated with it. Name is a string without spaces; price is a real number in dollars and cents; quantity is a non-negative whole number; weight is in whole pounds. Here are the Standard Pricing Rules for all purchased items:

1) Sales tax rate is 10% for clothing and electronics (with the exception of the below mentioned states).
2) Standard shipping cost is calculated by the following formula[1]: (20*(weight))*quantity.
3) Premium shipping incurs an additional charge of 20% over the standard shipping cost.
4) Assume that each item is shipped separately; i.e. don't combine grocery item A and grocery item B into a single package or with clothing item C.

You will be provided with a skeleton class code, that partially implements the driver class (or main), and also a skeleton for the base-class for all purchase items.

You will need to create derived subclasses for groceries, electronics and clothing from the purchase items base-class (you will be provided stubs for these as well).. Here are the special rules for those subclasses.

[1] Arbitrarily chosen formula for simplicity of programming.

- Groceries: Groceries are further classified as perishable or non-perishable. Perishable groceries require premium shipping. (Hint: Your derived class for groceries could have a field to indicate whether it is perishable or not.) Further, groceries incur no sales tax.
- Electronics: Electronics are taxed differently, based on the state to which they are delivered. The states[2] of TX, NM, VA, AZ, AK have no sales tax. Further, electronics may be fragile or non-fragile. Fragile electronics require premium shipping.
- Clothing: Standard rules apply. Premium shipping is not available.

Input and Output Requirements

For a shopping cart, a series of transactions are input from a given file. The input file name will be specified in the command line. The general format of a transaction is:

```
<operation> <category> <name> <price> <quantity> <weight>
     <optional field1> <optional field2>
```

Each transaction will be on a separate line. Each part of the transaction (i.e. the input values) will be delimited by one or more spaces between the fields.

Valid operations are: **insert, search, delete, update, and print**. For the insert operation, you will need to instantiate an object of the appropriate type (Groceries, Clothing or Electronics), and add it into an arraylist. For the other transaction operations you will search for name, delete an item based on its name, update the quantity of an item, or print the current contents of the shopping cart (for each item, it's: name, quantity, price after tax and shipping charges in ascending order by name) This is followed by the total charges for the shopping cart.

The insert transaction will appear as follows:

For clothing, it would look like

```
insert clothing shirt 20.50 1 1
```

For electronics, we also need to indicate if it is fragile (`F/NF`) and the state to ship it to. Hence, it would look like

```
insert electronics PS3 300 1 5.2 F NM
```

For groceries, we need to indicate perishable (`P/NP`) and hence it would look like

```
insert groceries cabbage 2.00 5 1 NP
```

The delete transaction would be of the form

`delete <name>` searches and deletes all objects with the name field that matches the given `<name>`.

[2] Note: States names are arbitrarily chosen. 2-letter state codes must be valid if specified.

Eg:

`delete cabbage` would delete all entries named cabbage, and output to the screen how many were deleted.

`search <name>` searches for all objects with name field as <name> and then outputs the number of objects found to the screen.

`update <name> <quantity>` updates the quantity field for the first occurrence of a matching `name`. Eg: `update cabbage 10` will change the first occurrence of an object called cabbage to 10, and then output the name and new quantity value for that object to the screen.

`print` will print the contents of the shopping cart in order by name, showing all attribute values for each object as well as the total charges for each. This is followed by the total charges for entire the shopping cart. Output of the shopping cart should be to the standard output stream (screen) and should be appropriately formatted and labeled for readability. This is not a complete shipping list, which would be handled separately by a different program.

Handling bad input data: You may NOT assume that the input transactions and fields are free from errors. For example, the input may contain commands that cannot be performed, which must be caught and reported without the program crashing or otherwise misbehaving (Eg: A bad operation value). Such transactions are to be reported, abandoned, and the program is then to proceed to the next transaction.

Design requirements

You will organize the shopping cart as an ArrayList of items (objects), sorted in ascending order by item name.

In each purchase item class you will implement a method calculatePrice() that returns the final price, for each of these items, based on the aforementioned rules.

Submission instructions:

Program Submission Instructions: Your code must be organized under a package called Assignment3 (note the case). Your main () method should be in a driver class called A3Driver.java. Zip the folder containing the source files (you will find it in your project directory under src. It will be called Assignment3), and submit it on CANVAS.

Design submission instructions: Your on-paper design is to submitted to CANVAS by the end of day on the designated due date. A workable design is worth 6 of the 20 points. If you miss this deadline you get a 0 on the design phase. 1 design per team.

The design of your program will require you to define and submit:

- A System level use case diagram
- A UML model of the needed classes and their relationships

- An ADT level description for each class that you create
- A functional block diagram showing the calling relationships between methods (library methods are not to be included; just yours)
- The algorithm needed for the driver logic (main method)

Analysis submission instructions: Write down on a piece of paper any questions that you have for me and any assumptions you are making. The questions should be primarily about your understanding of the problem to be solved. Submit your questions and assumptions to CANVAS by the end of day on the designated due date. 1 point..

Additional considerations:

- Coding standards - You must follow the class coding standards (see them on the class CANVAS page). These standards ensure your program is readable by others. Failure to follow these standards shall result in deductions on your assignment.
- Understandability - Comment your program so that its logic could be explained to your Grandmother (for example).
- Modifiability - We may be using your program again in future assignments, so design it for further adaptations and expansions.