PreLab3 Exercise

CS 106: Introduction to Data Structures

Goals

- Practice working with command line inputs
- Compare custom objects by implementing `Comparable` interface

Starter code location: /main/src/main/java/prelab

Review

Ever wondered what (`String[] args`) in the `main` method is? The `main` method is special but still has many similar behaviors as others you write. We can pass arguments (a.k.a. parameters) into the `main` method using command-line arguments.

In Eclipse, this can be done by `Run` → `Run Configurations…` → `arguments` tab. Enter the arguments you would like to pass into the `main` method, separated by spaces. The arguments will then exist in a `String` array called `args` in the order entered.

Processing Command-Line Arguments

Imagine three produce stands: one sells vegetables, one sells fruits, and one sells spices. The command-line arguments given will contain three tags, `-f` for fruit, `-v` for vegetable, and `-s` for spices. Each of these tags will be followed by two arguments: the name of the product and the total monetary value of the product currently in stock.

An example of a command-line argument might look something like this:

```
-f apple 10 -v kale 30.5 -v “sweet onion” 8 -s basil
```

If the above command-line arguments are passed, `args` would look like this:

```
{“-f”, “apple”, “10”, “-v”, “kale”, “30.5”, “-v”, “sweet onion”, “8”, “-s”, “basil”}
```

Our job is to process these arguments into different `ArrayLists` and create a `ProduceStand` object for each produce stand.

Implement the following in `Main.java`:

1.) Use a `for` loop to iterate over the product information in `args`.
   a.) Check for the tag (`-f`, `-v`, or `-s`)
      i.) add the product name in the appropriate `ArrayList` that is already initialized in the starter code.
      ii.) Add the value of the product to the appropriate `totalAsset` variable.
2.) Create a `ProduceStand` object for each type of produce.

a.) Pass the appropriate `ArrayList` and `totalAsset` variable to the constructor, along with a name of your choice for the produce stand.

b.) Print out each object using its `toString` method.

3.) Try running `Main` with the following command-line arguments:

a.) `-s salt 10 -f banana 3 -v lettuce 5 -f orange 9`

b.) You should get an output of something like this (only the fruit output is shown)

```
Steve's Fruit Stand is worth $12.0 and sells [banana, orange].
```

<u>Make the `ProduceStand` objects `Comparable`!</u>

To start, first, explore the `compareTo` method of `Strings`.

1. Run the following code in `main` method (you might have to re-type the quotes)

```
String orange = "orange";
String lemon = "lemon";
System.out.println("orange compared to lemon: " +
                    orange.compareTo(lemon));
System.out.println("lemon compared to orange: " +
                    lemon.compareTo(orange));
System.out.println("lemon compared to lemon: " +
                    lemon.compareTo(lemon));
```

2. Reason about the above output. *(hint: think alphabets and orders!)*

To enable `ProduceStand` comparison with other `ProduceStand`, do the following:

1. Declare that the class implements the `Comparable` interface
   a. The type of the other object being compared should be `ProduceStand` since we want to compare this `ProduceStand` to other `ProduceStands`
   b. Therefore, it would be `public class ProduceStand implements Comparable<ProduceStand>`
2. Implement the `compareTo()` method with the order specified as follows
   a. If this `ProduceStand` has a higher `totalAsset` than the other `ProduceStand`'s `totalAsset`, it should be considered "greater"
   b. If the `ProduceStands` have the same `totalAsset`, then the one with the earlier lexicographical `standName` should be "greater" than the other.
   c. If the `ProduceStands` have the same `totalAsset` and `standName`, then the object with more products is "greater".
   d. All of the above are the same, their comparison is "zero".
3. Print the result of the three comparisons by comparing two different `ProduceStands` each time. Use some command-line arguments of your choice.

Takeaways (fill in the blanks)

____ is just a String array containing command-line arguments

- Command-line arguments are separated by a _____
- If there is a space in the argument, they must be wrapped in ________ ______

__________ allows comparison of customs objects with some specified order.