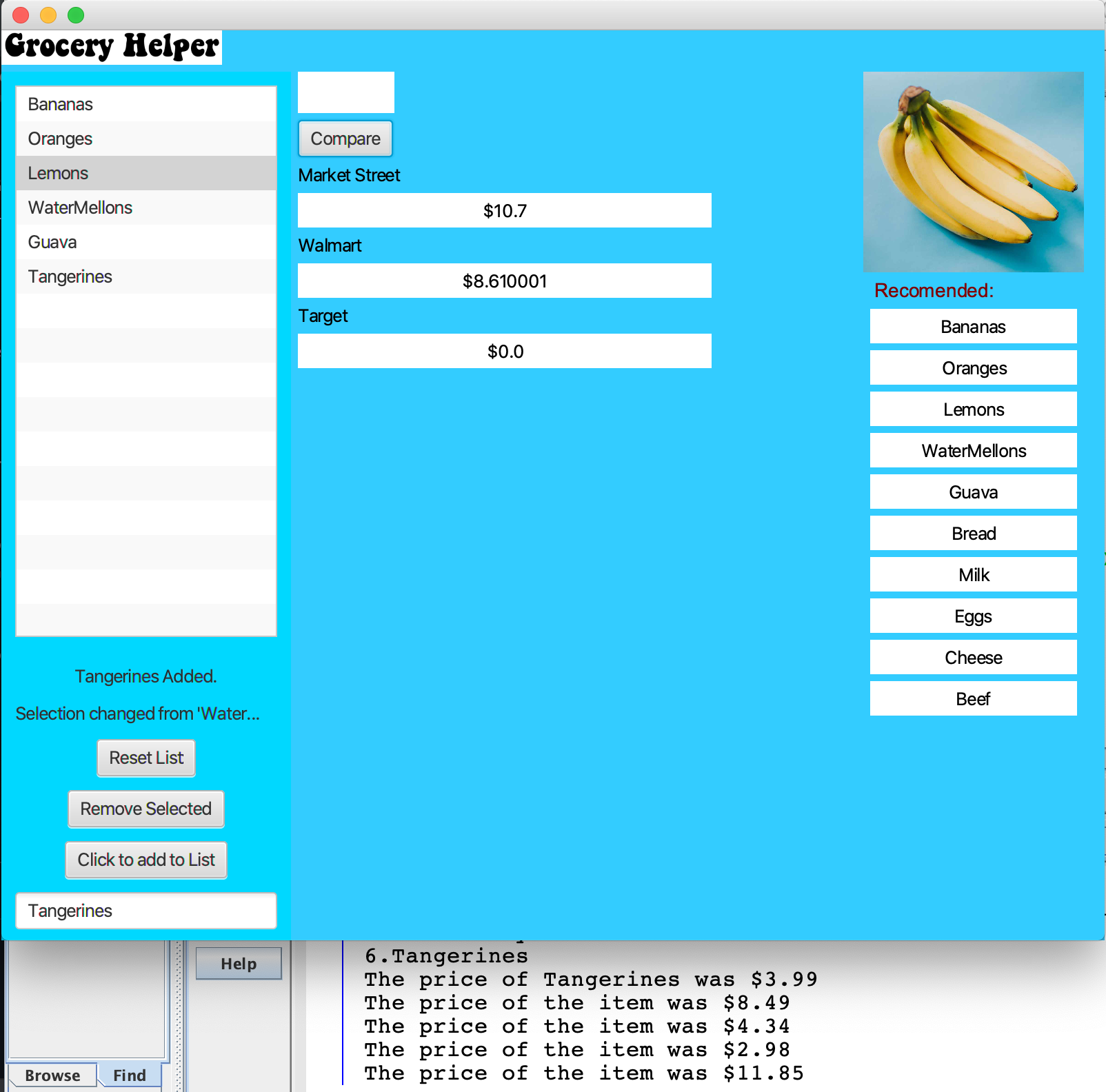
# Saif Chowdhury Tucker Hortman Brian Michelle Jeff Tessitore

# **Shopping Scraper**

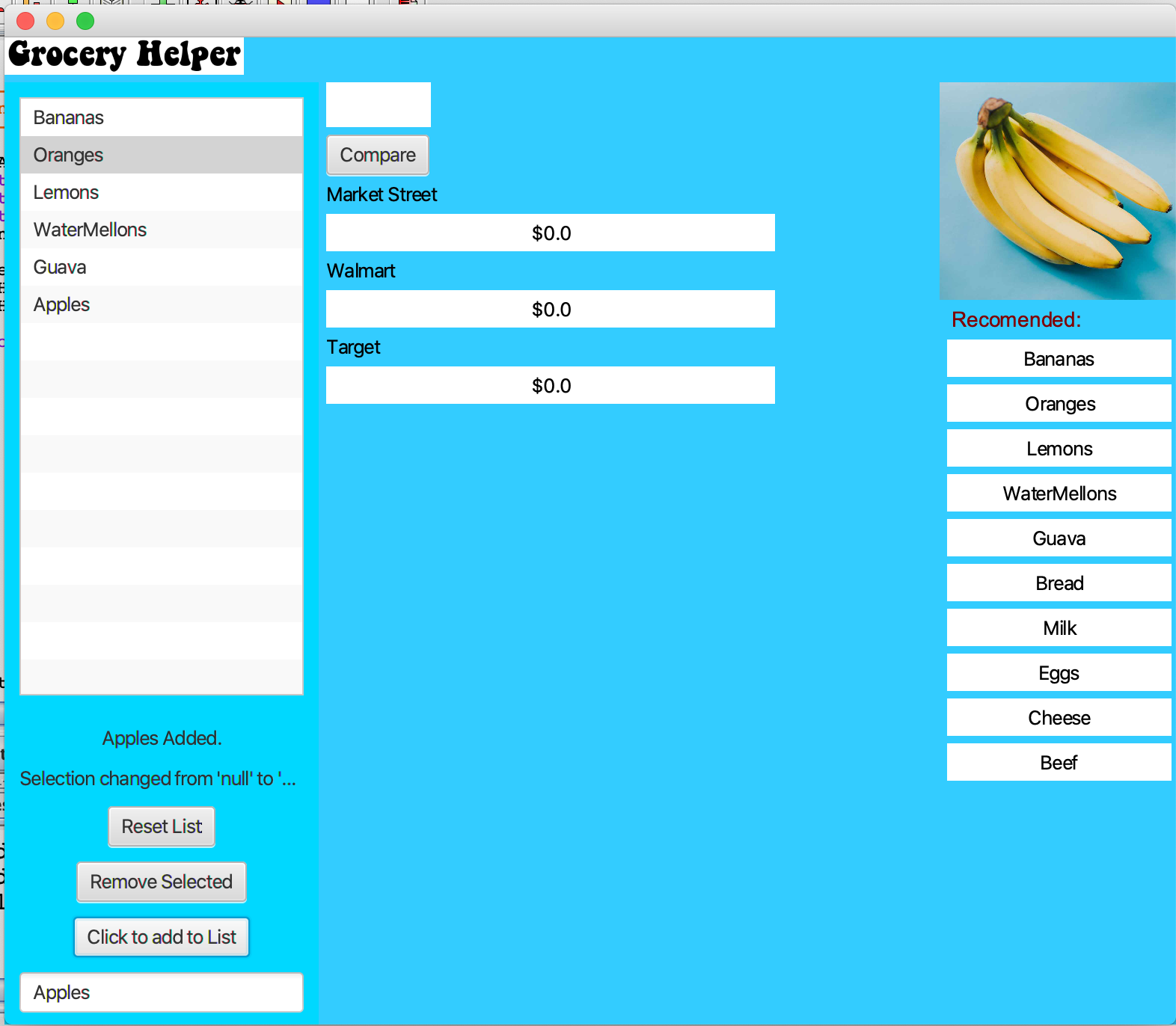
Hello! This is our Grocery Shopper App!  
To run this, simply open up Grocery\_UI\_App in your favorite IDE and hit run. Download this whole folder if you haven't, then run the JavaFX App to use our prototype. If you just want a demonstration of the scrappers, similarly open main.



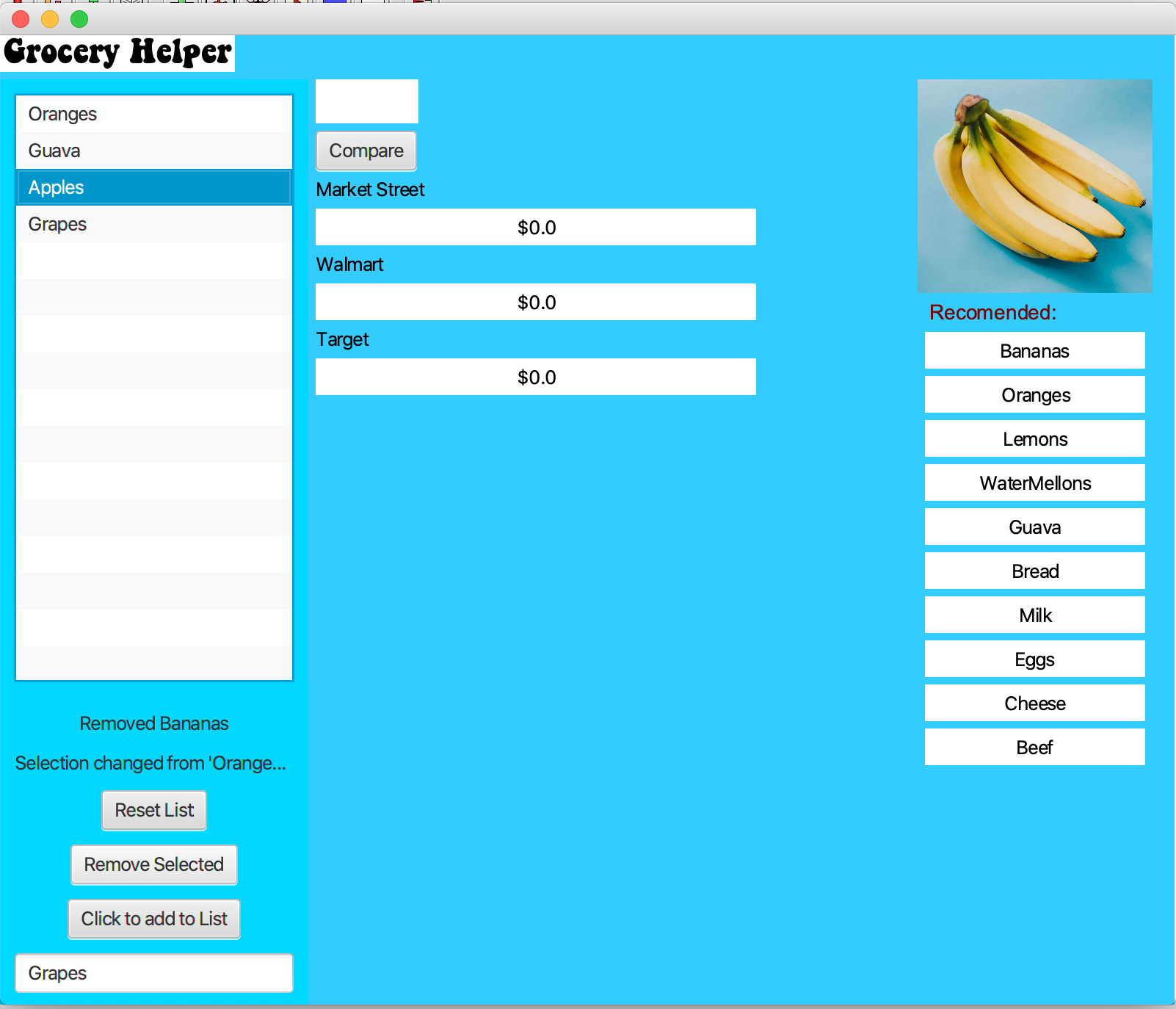
Currently: You can add, remove, and reset list.

Here’s A Picture of These In Action:

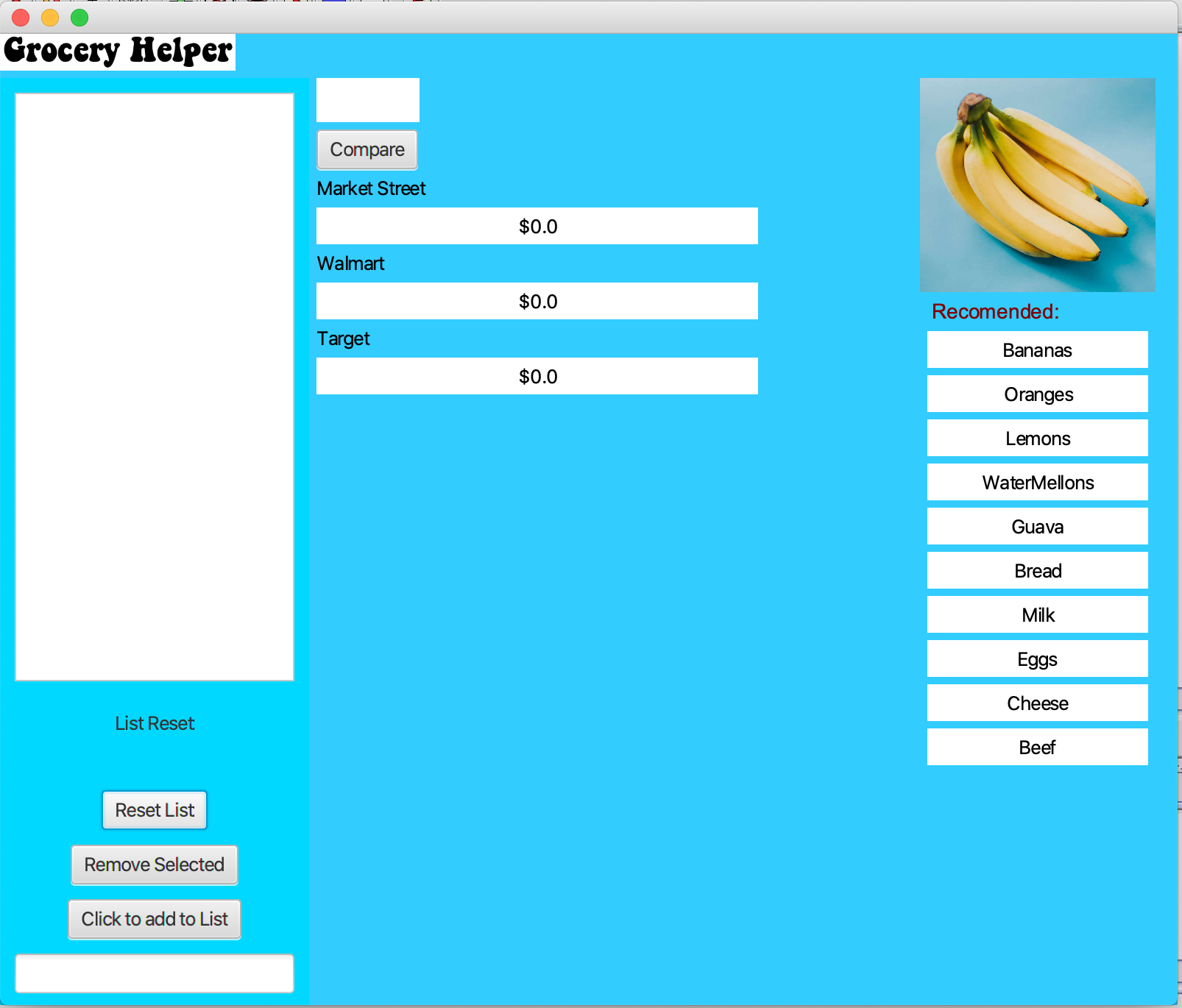
Add:



Remove:



Reset:

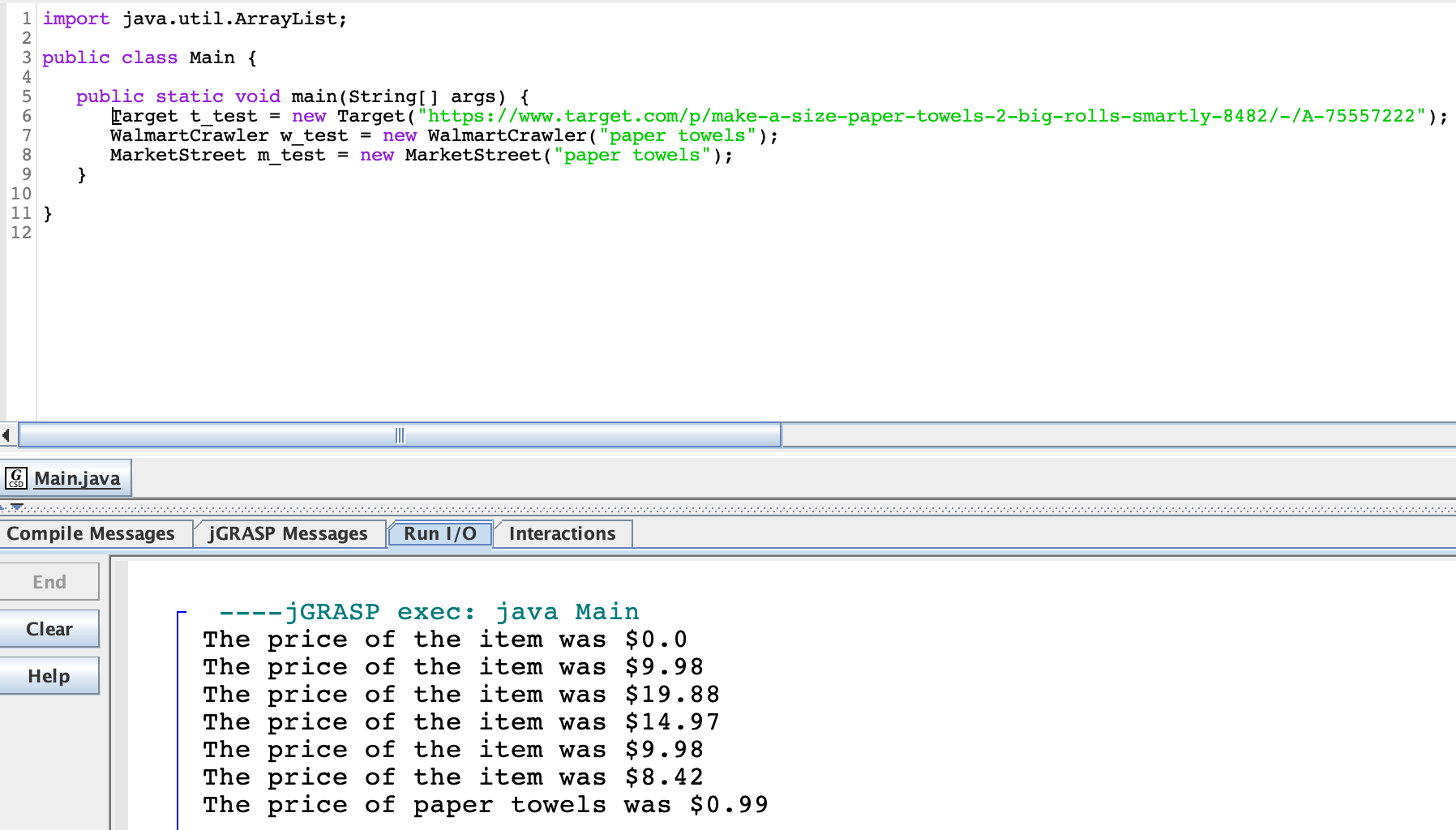


You can also hit the compare buttom and our interface will automatically run the scrapper programs necessary to find the prices online for MarketStreet and Walmart.

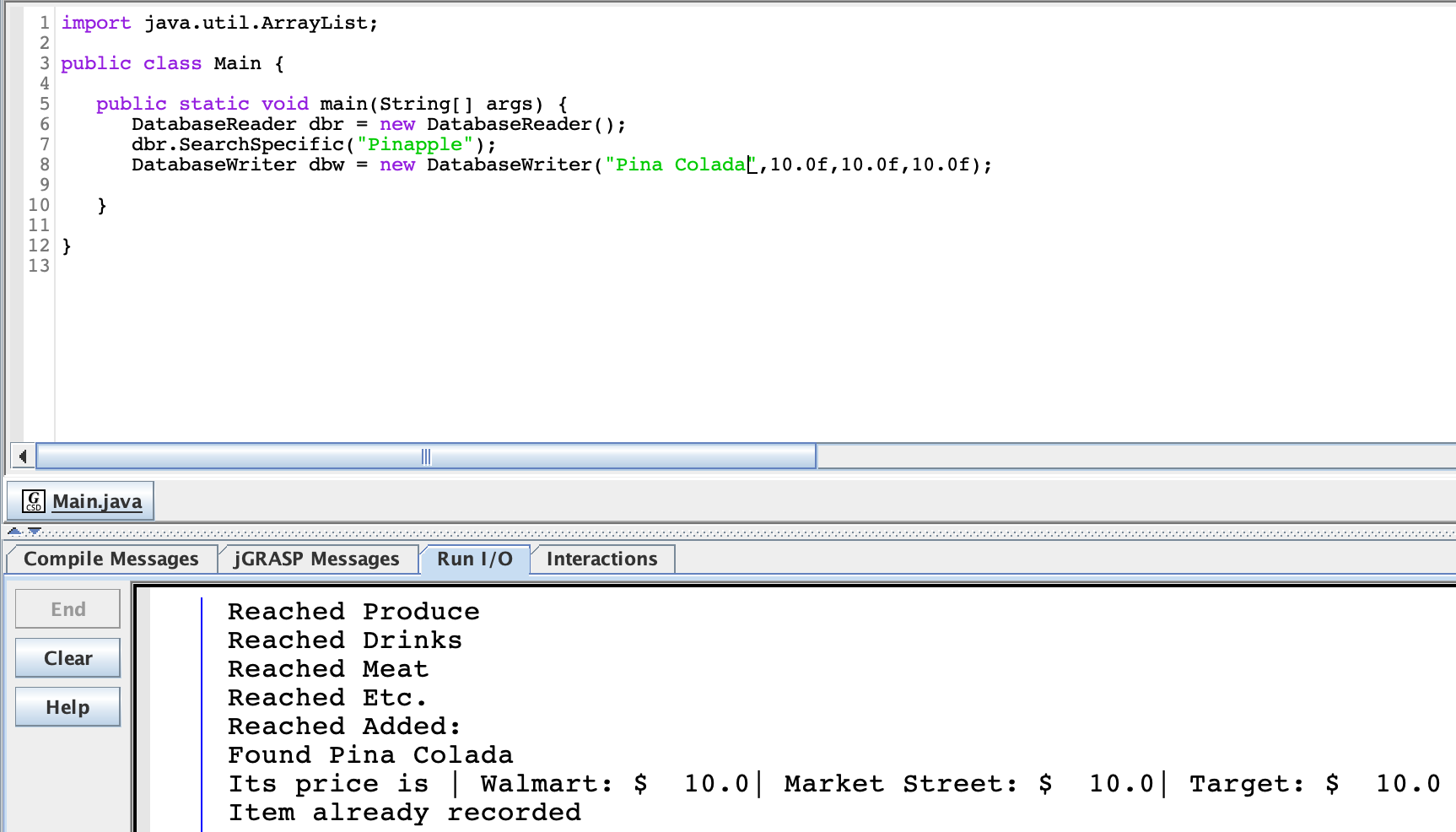
And it will add the totals of each store, if the price was not already found in the database. In which case it will not run the scraper, and will instead add from the database.

Combining these features really improves speed and versatility.

Bellow you will find our HTML Scrapers In Action.



This is an example of the Database Reader that was later added to store the values of previously searched items:



Heres a Photo of Our Database:  
Each of us found one of the 4 Primary lists in this database. We then added the functionality of the growing database in the "Added" Section.  
  
Additionally, at this time, while we get our Target Scraper to work, in order to use Target, a maintainer is required, a second person, who will go into the database.txt, and manually enter in the prices for Target for comparison.

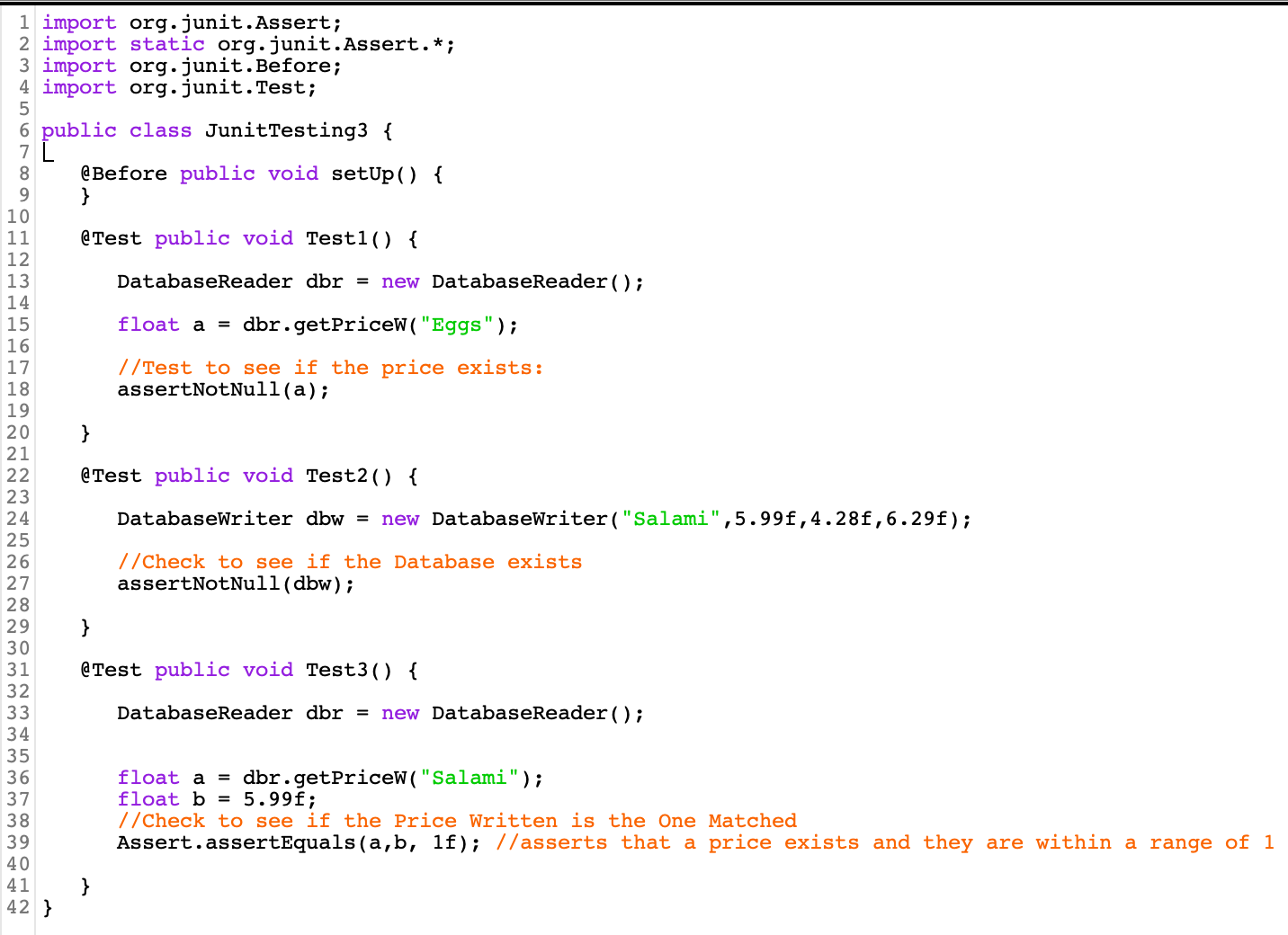
As you can see, the more you use this app, the better it will get. Each time you enter in an item, the software scrapes the internet for its prices. These prices will be recorded, allowing you to have an ever expanding list. Right now the size of the list is limited to an array of size 500.

In the future, we'd hope to expand this by allowing past prices to be updated, and increases the number of stores.

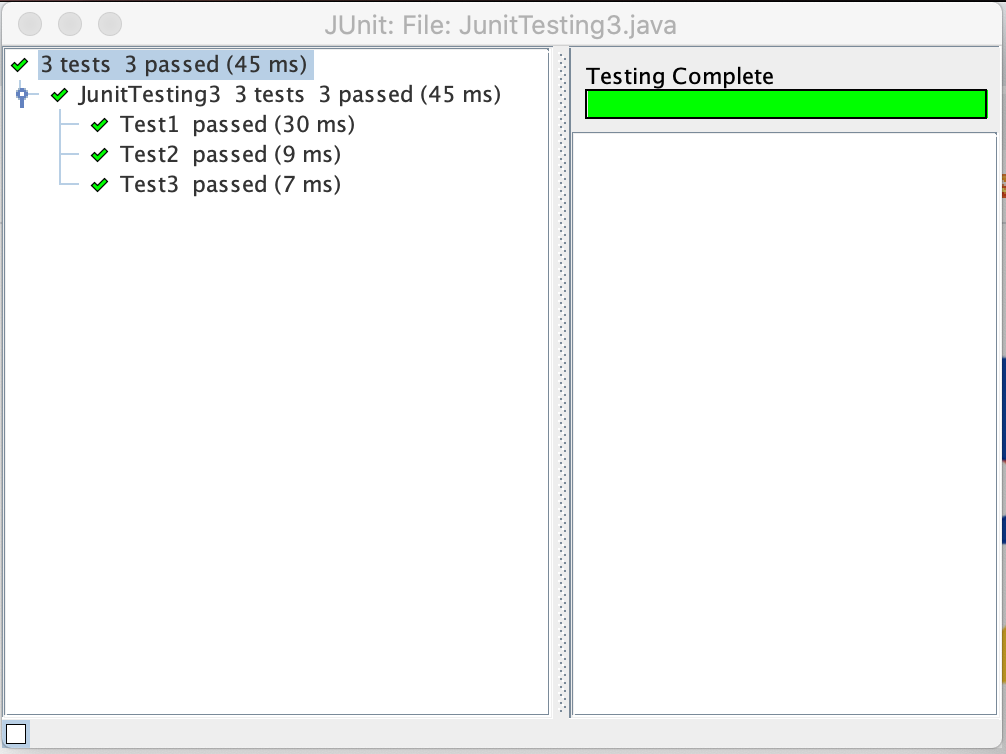


We hope you have enjoyed this demonstration of our Grocery Comparison App!

Additionally, here is an implementation of Unit Testing:



As you can see, it passed:



We chose to focus on 3 key areas in the process of making these JUnit Tests.

1) we make sure that there is a number value associated with the String Eggs. Which is done by sending string Eggs through a method to enter in the value to compare against a large number of strings, each matched with a line in that array. Each item in that holds the string that holds the item name. Before it makes its way back to us, it needs to see if the value Eggs is even present. For if it were not present, that Writer would create an entry.

The Database Reader gets text from a file database.txt. Because in this case, an Entry for eggs has already been entered in, the Test case for asserting a number for Eggs exists for the store Walmart is true, passes.

2) secondly, we chose to covered, the matter of initializing the database writer. For if the writer object, with the constructor properly working, hadn't initialized, then we would not have been able to access it for when we need to add a new set of values into it.

3) our third test covers checking if the price an item returns, is what it needs to return. This is very important to make sure than somewhere along the process of coding these three stores, they didn’t get mixed up. It is very easy for one to overlook and swap items mistakenly somewhere along the way. Including a unit test which confirms none of the prices got swapped at any point, really helps us to confirm that the whole thing is working and can be trusted as a credible source of comparing prices.