Report 1 Final Version

Title: Grocery Buddy

Authors: Saif Chowdhury Tucker Hortman Brian Michell Jeff Tessitore

Purpose: To build an application that helps students in their daily lives.

Scope:

Our project is going to let you type in your desired shopping cart into an app that will then tell you the cheapest option, factoring in gas. The app will use perpetually updating data to get the prices of various grocery stores and supermarkets. Every time a user enters in an item, the name of that item will be stored. The program will automatically be able to do a Google search and find the prices of this item at each of the stores. The program will then add the total of all these items and let the user know which of these stores are the best to choose from, based on the items listed. If an item is not available at a store, the user will be notified. However, the presented comparison of prices will be based on the shared items among stores.

This is primarily used for grocery items to help college students find the best prices. However, its applications allow for it to find the best prices of virtually any item shared among multiple stores.

There are two ways to implement the calculation of the distances to find nearby stores. The first is to use GPS information from Google maps to collect the distance from any given location. The second is to create a data set (table) that includes all the distances to grocery stores commonly used in a given area. For the purposes of this project that area we will focus on will be a central point in Lubbock, such as Bledsoe Hall at Texas Tech University. This can later be expanded to include any location the user desires.

If we go the route of one central location, we will first need to list the grocery stores we will be searching for. For example, some common grocery stores in the Lubbock area include:

* Walmart - Marsha Sharp
* HEB
* Target
* Market Street
* Food King

This can later be expanded upon. All a grocery store needs to qualify is a list of their products on their website, which most grocery stores have. (Maybe we can implement code that searches stores tagged as “grocery store” and then checks their website to see if there is a “cart” available or something similar.)

However, if we were to later include any city of choice for the user, we would first need to implement a software that lists the names of all stores that sell groceries in that area based on location. This can also be done by leveraging data collected from Google.

Top Five Features:

* HTML Parser and Scraper
* Adding/Deleting items from a list
* Comparing prices and ignoring unshared ones
* Database that holds shop prices, and locations (we may tweaking this feature)
* Display/Intuitive Design (making everything look pretty and fun)

Developing Feature #1:

For the purpose of the first top feature, we decided to go with the HTML Scraper. We did this because we recognized this would be the biggest hurdle in our project.

So we began with research of various sites nearby our area in Lubbock, Texas. After learning about the HTML structure of each of our respective pages, we then developed a scraping program that was capable of finding the price of that specific store. It needed to be done this way because each store has different hierarchies and classes and tags, that they use to organize their store prices.

However, we believe that in learning how each of these websites do function, we will be able to find a common system that allows us to get prices from any page, eventually.

Each website needed its own system for finding its prices essentially. So each of us was tasked with Scraping a store, Walmart, Target, HEB, and Market Street.

Walmart: Brian showed us about how, if you right-click on the price of a website and hit inspect element you can find exactly where in the HTML, that element is located. This was a super helpful tip.

He also first introduced a useful HTML scraper for Walmart, a program that we could reconfigure to get the right data from our websites as well. We are using GitHub to keep all this information organized. This includes our scrapers and all the auxiliary programs that utilize these.

Market Street: Utilizing the code that was made by Brian, I was able to reconfigure this to work for Market Street. The key difference was Market Street requires cookie authentication of the location to access prices. So after inspecting elements, I was able to find several cookies. With help, I was able to get these cookies added to the get URL. A few minor modifications were needed (such as strings that held the element attributes and tag names) but essentially the MarkStreet class holds the same float value for the item price.

It holds the first item on the site so I’m really relying on Market Street’s lowest price listing. This is fine because we are only looking for the cheapest possible product from every website.

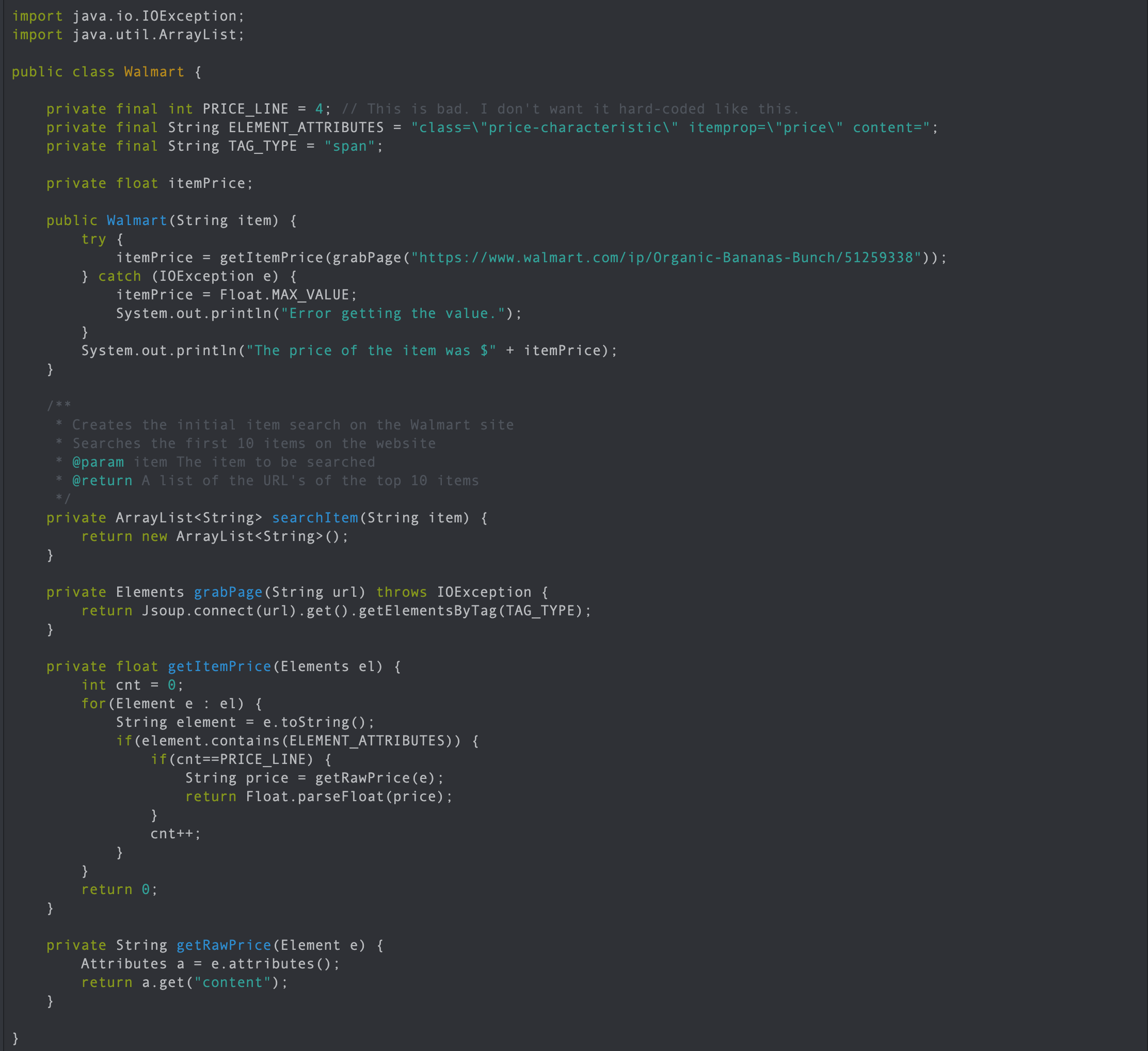
Target: We were having a particularly difficult time with Target because essentially, the information for the price just wouldn’t show up in the HTML was retrieved. The price could be visible from the inspector, however. We suspected this had something to do with the cookies of the page again. Eventually, we were able to find the solution to this problem.

HEB: early on, we realized that HEB had a “Not a Robot” Captcha system. We realized this was going to make searching through HEB probably impossible because essentially we are designing a robot to do the searches for us. So we decided to pivot.

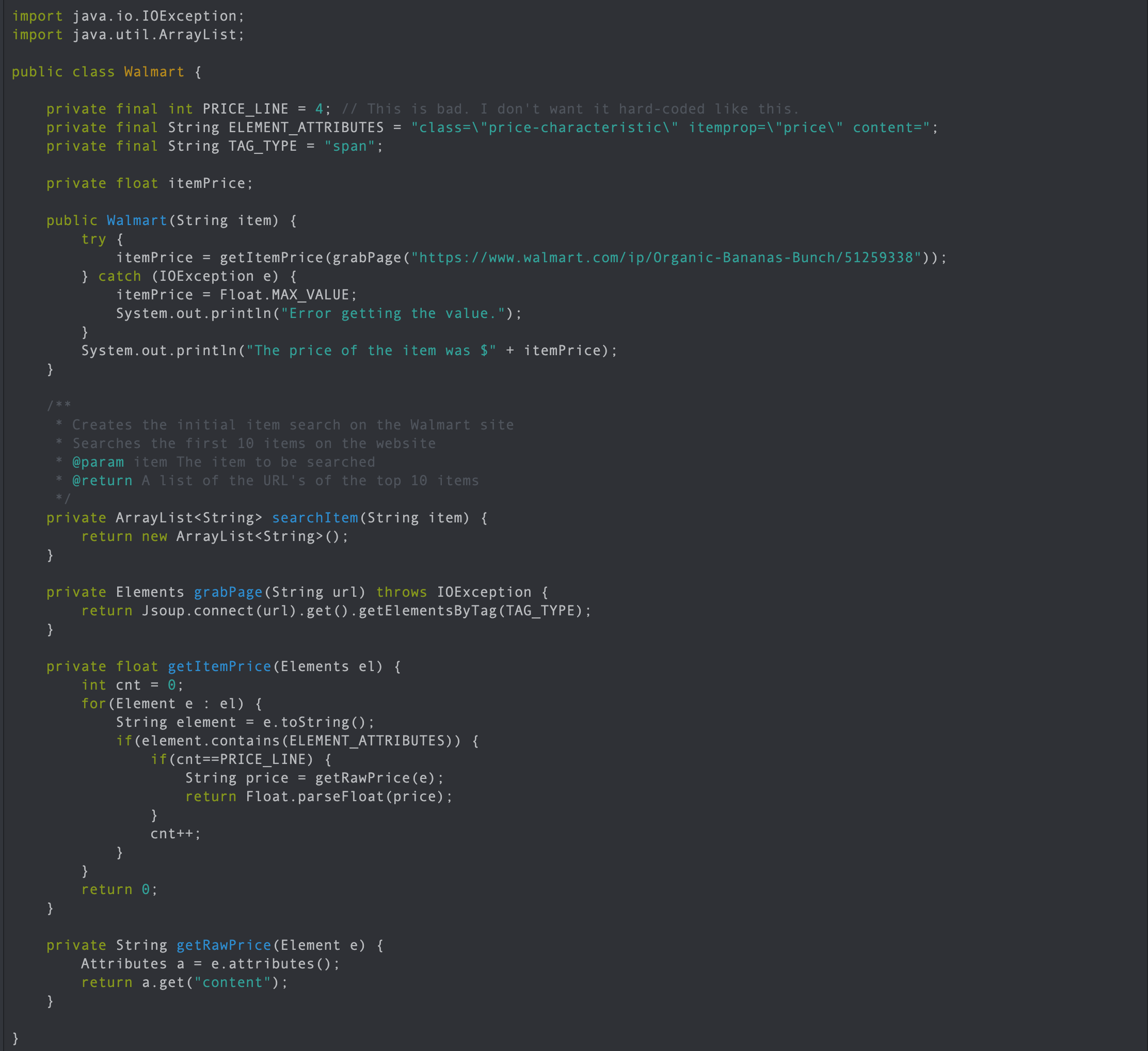
It is commonly recognized that HEB is more expensive than other stores. For this reason, we believe it not to be an incredible disservice to not include. Essentially our application is a price comparator.

As long as we have more than one price to compare, we are achieving our goal.

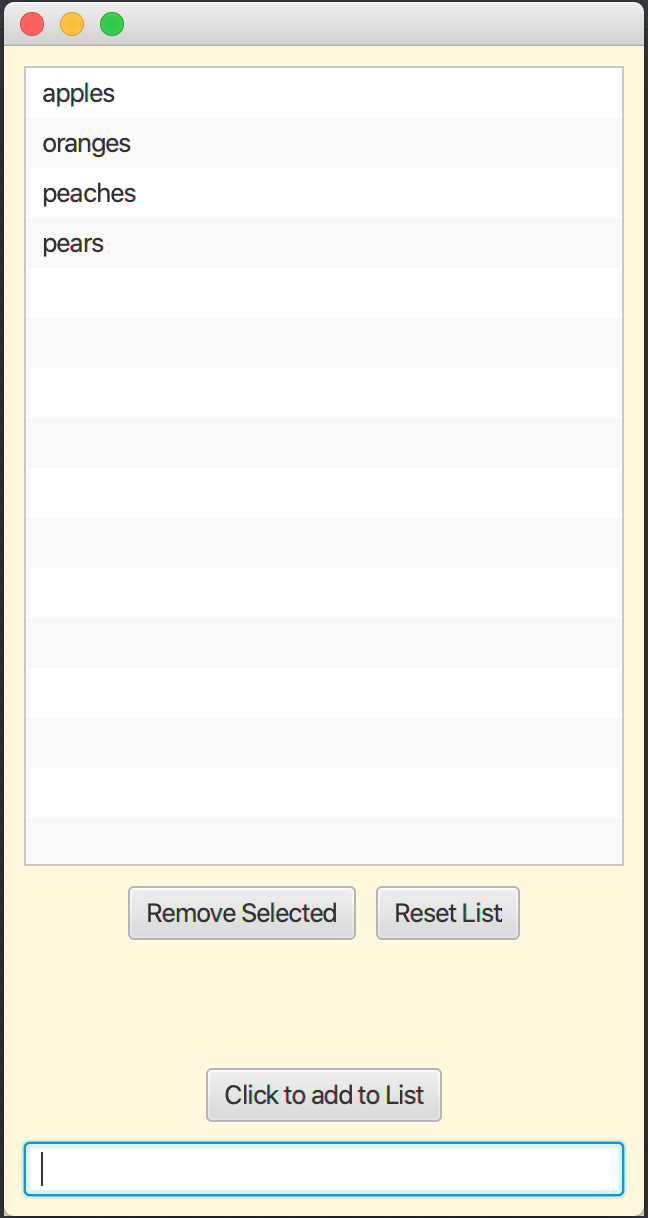
The Code that enabled us to move forward:



Continuation of the Walmart Class:



Future Goals: Bellow is a prototype of the GUI interface we hope to implement which lets the user enter in his list of items to be compared.



Example of what Saif’s Market Street can do utilizing this code:

We achieved similar results in the other stores

At the bottom, you will see we are able to get the price of any item in their catalog by only entering their String. The main drawback I see with this is the speed it takes to get this information. That is why I proposed the database that holds these values for later retrieval.

