Technical documentation

Website word counter

Jack Oporto

October 18, 2020

Professor Al Kafaf

**Summary:**

This program demonstrates the use of JavaFX to layer on a UI to control the function of this project: to **output the top 20 words and their frequencies**.

The UI gives the user two options: Enter their own URL or select the default of this project: The Raven by Edgar Allen Poe.

This program utilizes jsoup to scrape data from a given URL or the default.

**Implementation:**

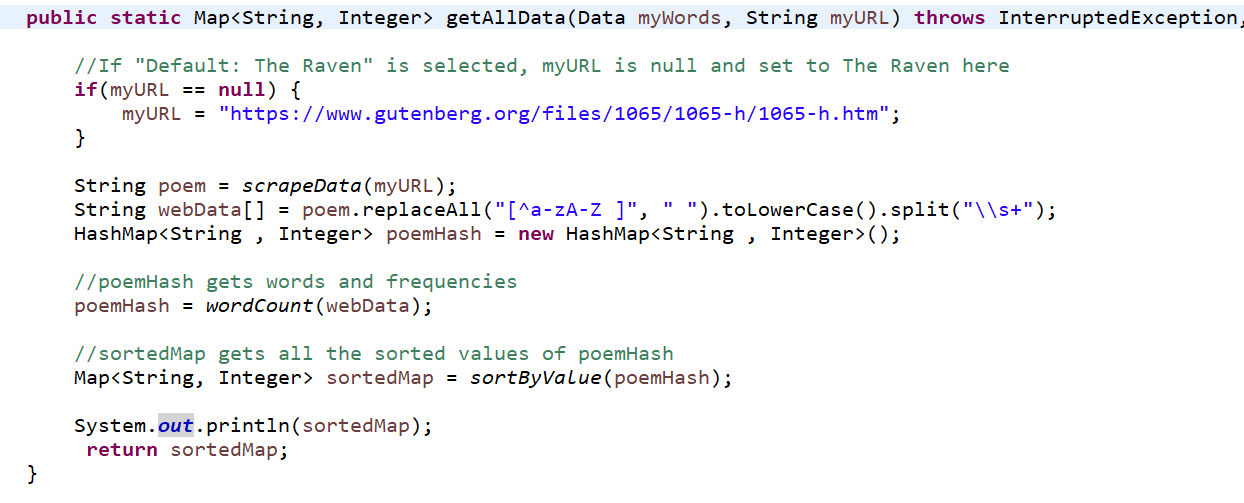
Since I knew we’d be building on this program over the course of the semester, I knew from the start that I’d want to extend the functionality to include other website URLs. This project works best with our given website with The Raven, as other websites are formatted differently, and my code doesn’t account for words not inside given html parameters.

First, I got everything out of main and moved it to launch() so my code would process at the same time and have access to all my data.

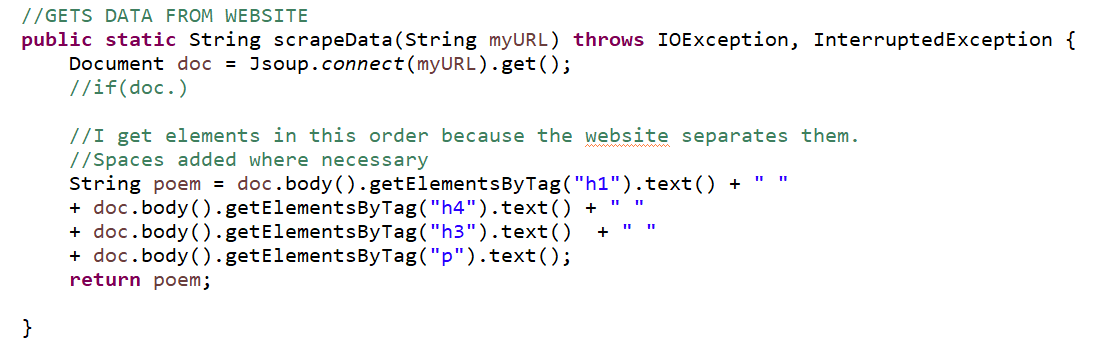
JavaFX allows for us to create multiple stages to transition to and from and I used them to transition from my initial prompt to my output.

**Functions:**

I separated the processing of my data into different functions to make things easier to manage, and used **one final function to call them all, returning the final Map containing my list of words and frequencies, in order**. This way, JavaFX’s built in systems can recognize my data and display it correctly.



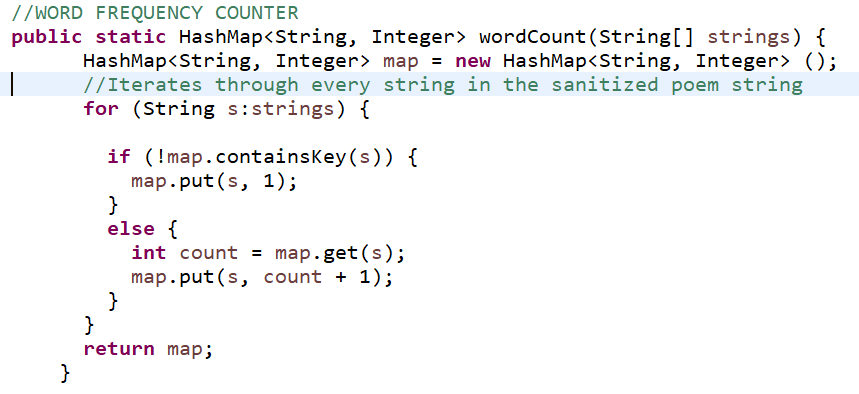
getAllData() is our main data processing function. It’ll call the functions to **scrape the data from the website, count the frequency of each word then sort it**. In addition, it cleans up the data before processing of any punctuation as well as standardizing every word to lower case for proper processing.



scrapeData() uses Jsoup to connect to the url given from getAllData()

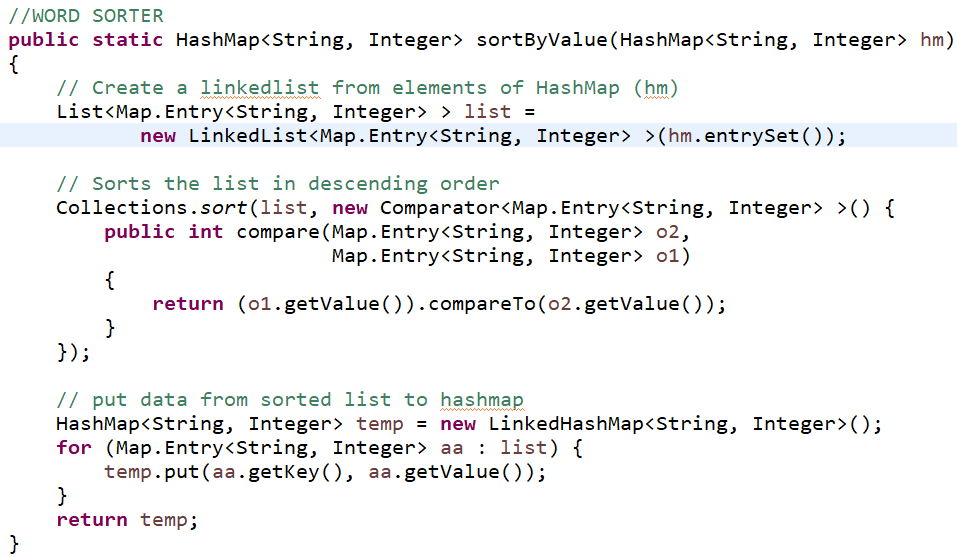
Jsoup allows us to get data by tag and that was the more explicit manner to scrape the data from this website.

It ends by returning the entirety of the poem as a string.

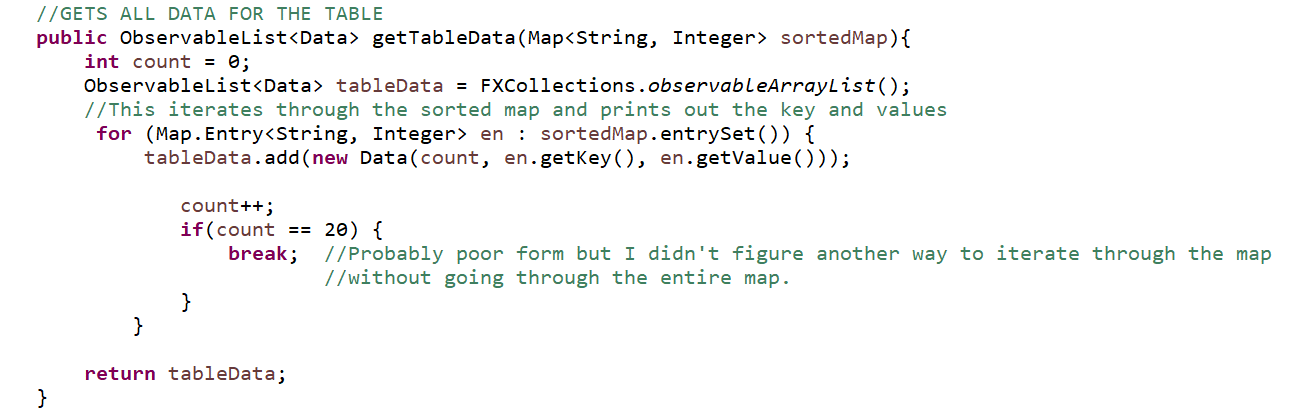


Using a hashmap, this loop will iterate through the entire string and place each UNIQUE word as a KEY in the map, and each time it appears again, increment the VALUE.

This returns that map of data.



Using the hashmap of data from wordcount(), sortByValue reorganizes the data such that it is organized by value, starting with the highest integer value.



Most importantly, this JavaFX object called ObservableList accepts the sortedMap and plugs it into a TableView, another JavaFX tool for displaying data. This results in the final output, showing each word, their frequency and their index in line.