**Google Search Engine Simulator**

**The First Programming Assignment (PA-1)**

Ahror Abdulhamidov

CS-146, Instructor: Dr. Mike Wu

San Jose State University

October 23rd, 2020

**Design and Implementation**

This application simulates Google Search engine. It is a GUI based application that uses a web crawler to search for URL page links, uses heap sort to sort the list of pages by their ranks, builds a max-heap priority queue, and provides with the priority queue associated features.

**Front-end**

The front-end is GUI based. The application launches from the Main class, which calls the launch method from the GUIForm class. When launched, the application shows a window with a search field, a big text panel, and 6 buttons, each associated with a feature of this application.

Use Cases:

1) User will be able to type a key word into the Search Field and click Search button to search through web. Immediately, the application will show the entire list of 30 page links associated with the key word, ordered in ascending order by their page ranks, each having randomly page ranks. There will be 5 ranks: frequency of the keyword, website's age, linked from other websites, paid advertisement, and the total rank, which is a sum of 4 other ranks.

2) After displaying the page links, the user will be able to make a priority queue from the current page list. To do that, user will have to press the Make Priority Queue button. Immediately, an input dialog will pop up, asking the user to type the desired size of the priority queue, which cannot be greater than the size of the current page list. After setting the heap size of the priority queue, the dialog will close, and the page list will be updated in the text panel.

3) User will also be able to add a new page by clicking the Add Page button, which will show input dialogs, asking for the page URL and the 4 rankings. After it’s done, the program will use max heap insert to insert the new page to the page list, and it will update the text panel.

4) In addition, user will also be able to increase page rank by changing the 4 page rankings for a chosen page. User will have to click the Increase Rank button, which will show dialogs asking the number of the page (index), and new 4 rankings. After that, the program will use max heap increase key on that page and will update the page list on the text pane.

5) If the user desires to extract the highest ranked page, he will be able to do so by clicking the Extract Best Page button, which will use the heap extract maximum method to remove the highest ranked page from the list, and then it will show it in a pop up dialog. If the user does not want it to be removed, the Show Best Page button would be sufficient.

**Back-end**

The backend mainly contains 3 classes: MaxHeapPriorityPageList, Page, and WebCrawler. Both of them are used by the GUIForm.java class, which is responsible for the front-end.

At its heart, is the WebCrawler class, which is responsible for crawling through the web to find relevant pages for a specific key word. The main attribute of that class is the array list of URLs, and the main two methods are the search method, which searches for the links by the key word, and the getUrls method, which returns the array list of URLs. The web crawler was initially provided by the instructor, but was slightly modified, so that it contains longer URL links (still not the entire link) and array list of links instead of hash set.

The Page class is the inner class inside the MaxHeapPriorityPageList class, which describes each page. The properties are: page link (String), 4 ranks (int), total rank (int), and a randomizer variable. Every new Page is created with the specified page link (URL). Whenever a new Page object is created, random numbers from 1 to 100 are assigned to the 4 ranks of that page, and the total rank is calculated. The class has some getter/setter methods, calculateTotalRank method, which calculates the total rank of the page, and toString method that is usually used for displaying the Page info.

The MaxHeapPriorityList class is the largest by the number of methods class. It contains 3 properties: A[ ] – array list of Pages, int size – size of the entire array list, int heapSize – size of the heap inside the array list, and later also the size of the priority queue, which is part of the array list. The class’s constructor takes an array list of String URLs, makes Pages out these URLs, and stores them in the class’s array list of pages. The class also has a swap method, which swaps 2 specified Pages in the array list, and 2 toString methods, for the entire list and for the priority queue. After these methods, there are 2 sets of methods: first set of methods is related to heap sort, and the second – to priority queues.

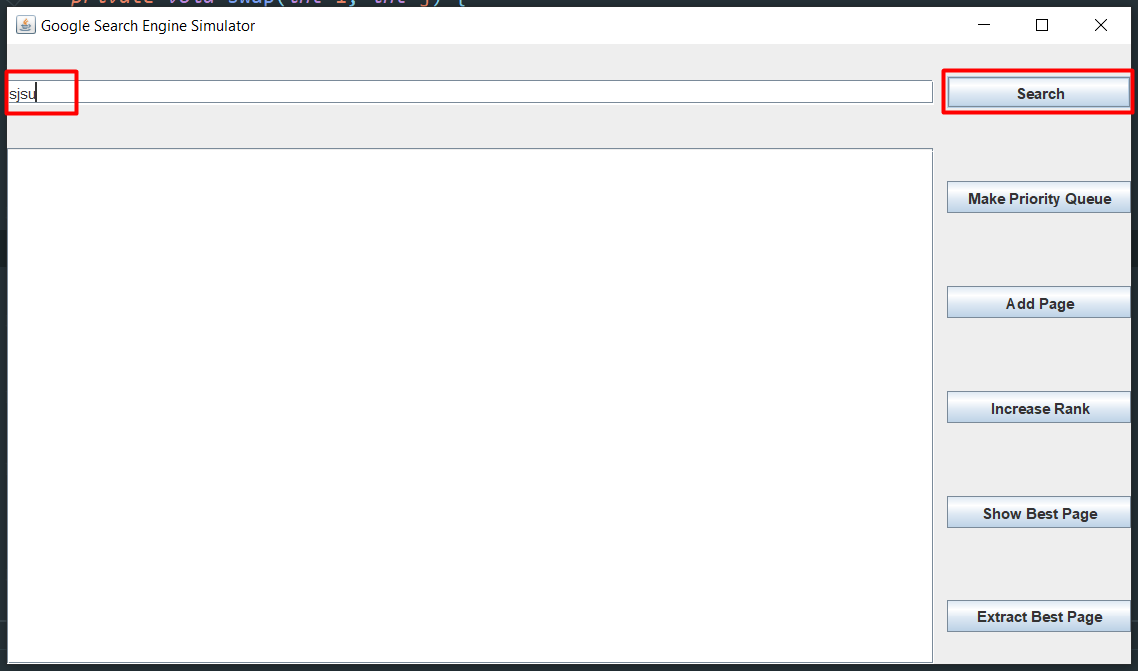
Among the methods for heap sort are: helper methods, maxHeapify, buildMaxHeap, and heapSort. The helper methods are: left(i) which returns the left child of i-th element, right(i) – right child, parent(i) – parent node. The maxHeapify(i) method makes the i-th element fit the Max Heap property (parent is larger than its child) in the array list. The method uses totalRanks to compare pages. The buildMaxHeap() builds a Max Heap Tree out of the array list of pages.

Among the methods for max heap priority queue are: makePriorityQueueOfSize, heapMaximum, heapExtractMaximum, heapIncreaseKey, maxHeapInsert. The method makePriorityQueueOfSize(heapSize) makes a priority queue inside the sorted array list, simply sets the heapSize with the specified number. The method heapMaximum() returns Page with the highest totalRank. The method heapextractMaximum() removes the highest ranked Page from the priority queue, restores max heap tree, and returns the deleted Page. The method heapIncreaseKey(i, 4 ranks) increases the total rank of the specified Page by setting new specified ranks, and rearranges its position in the max heap tree. The method maxHeapInsert(link, 4 ranks) inserts a new Page to the max heap queue.

**Testing Screenshots**

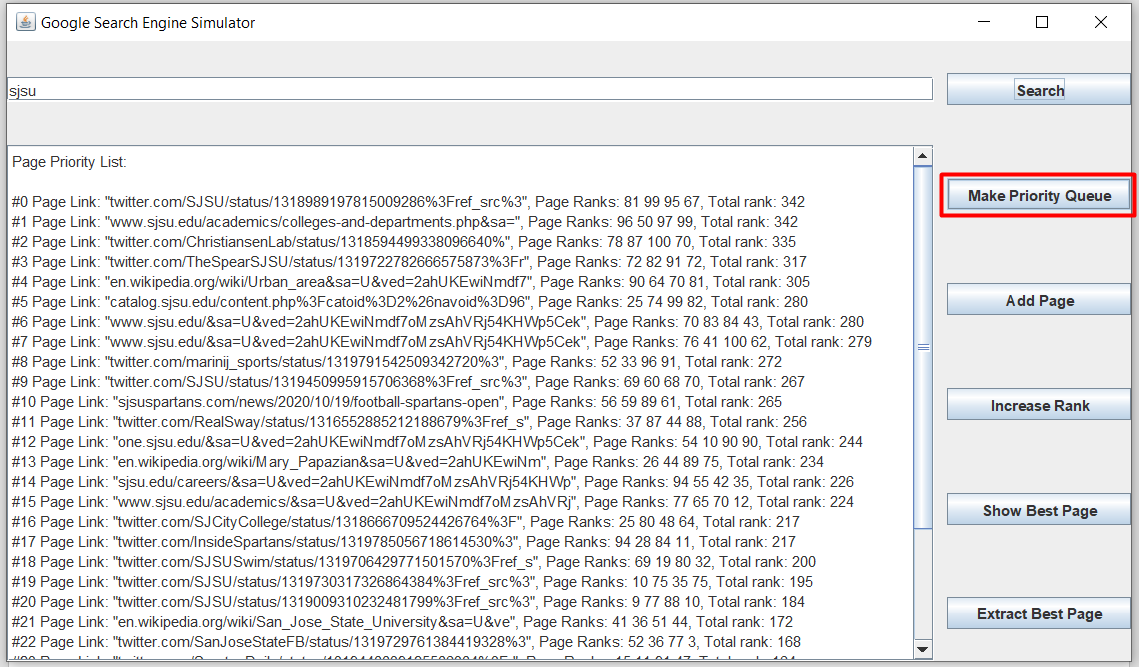
Use Cases (Specified in the Front-end section):

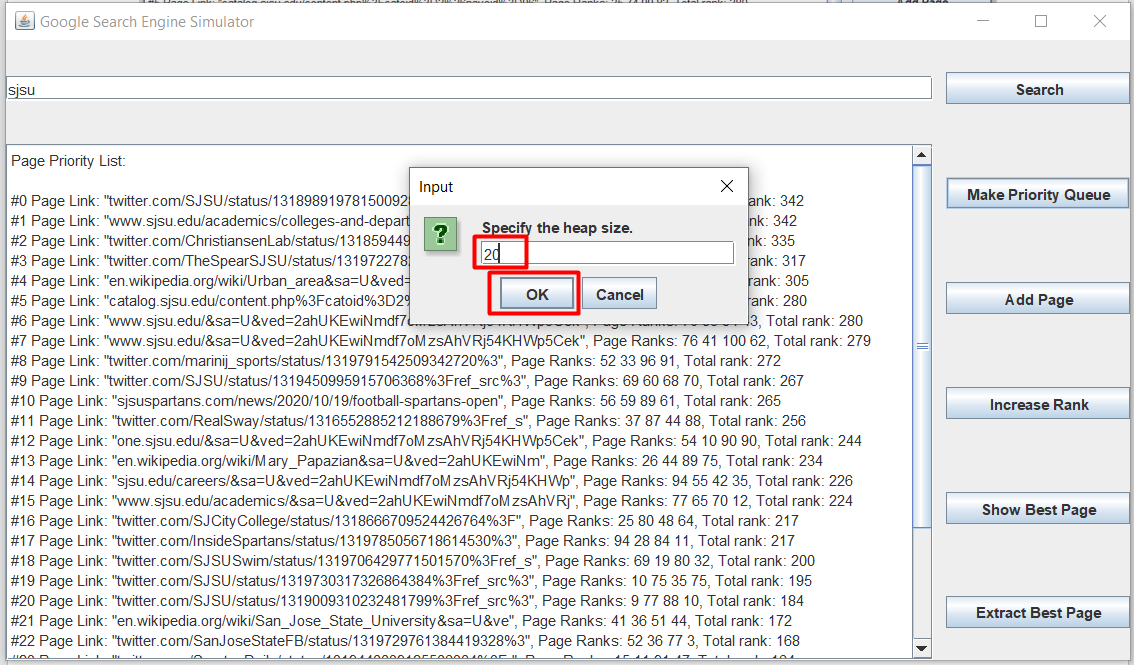
1)

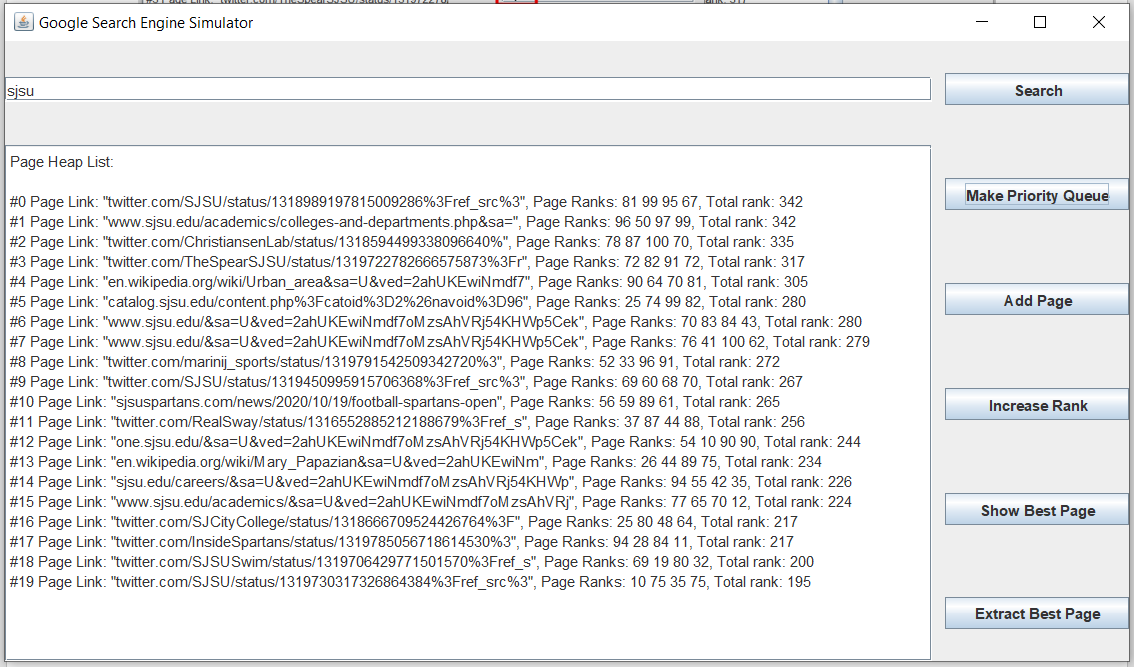




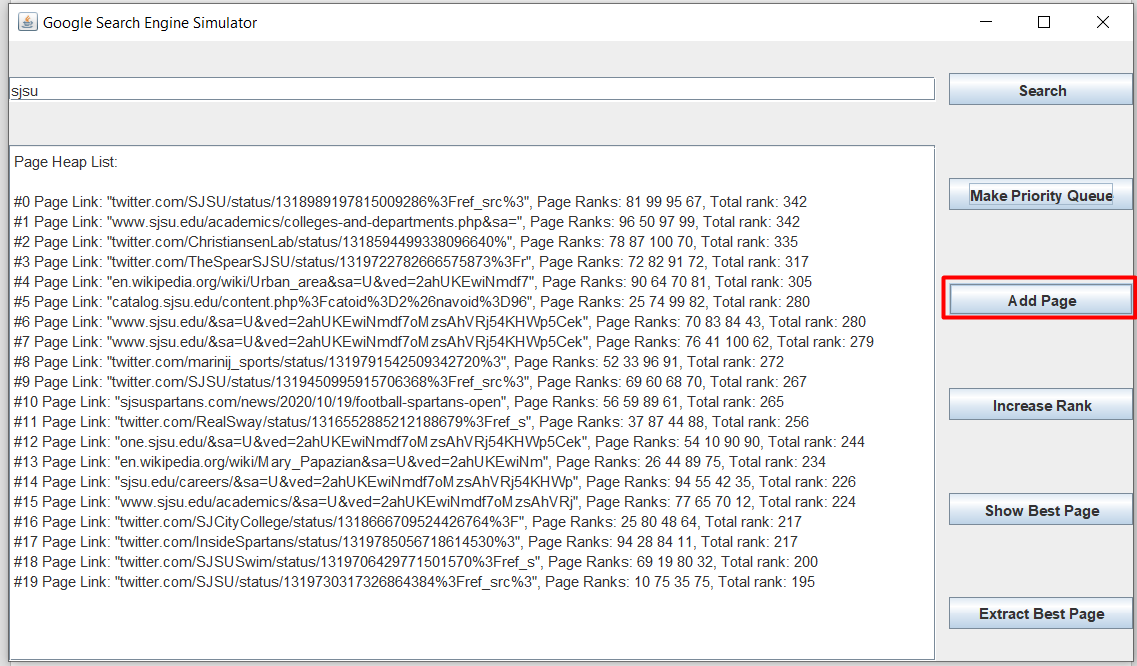
2)

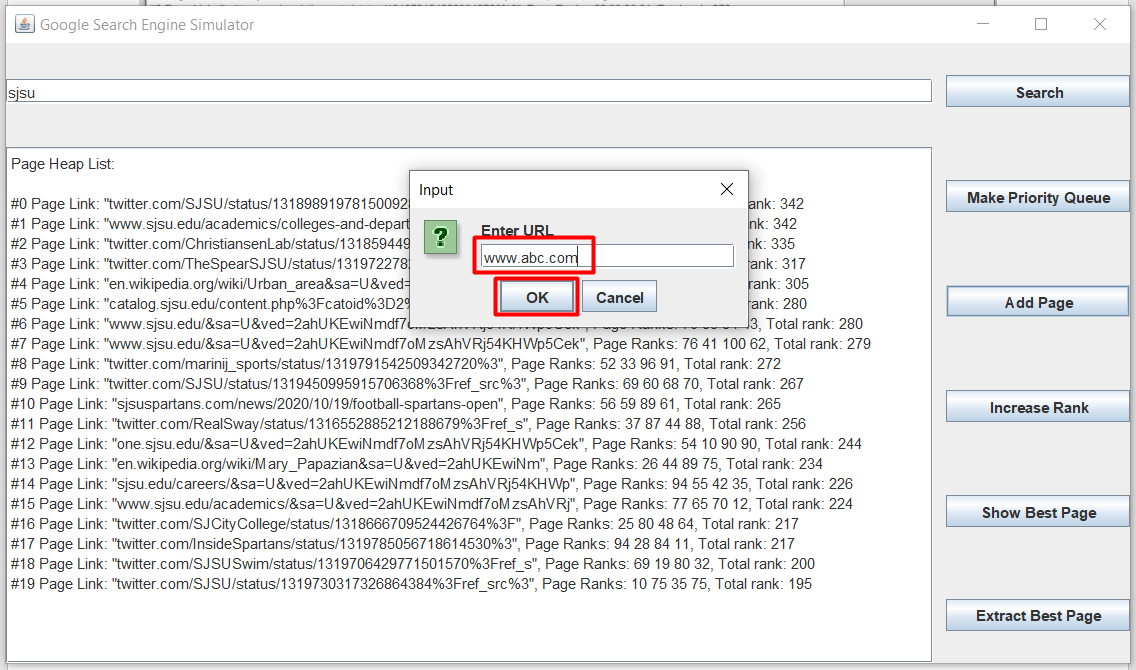


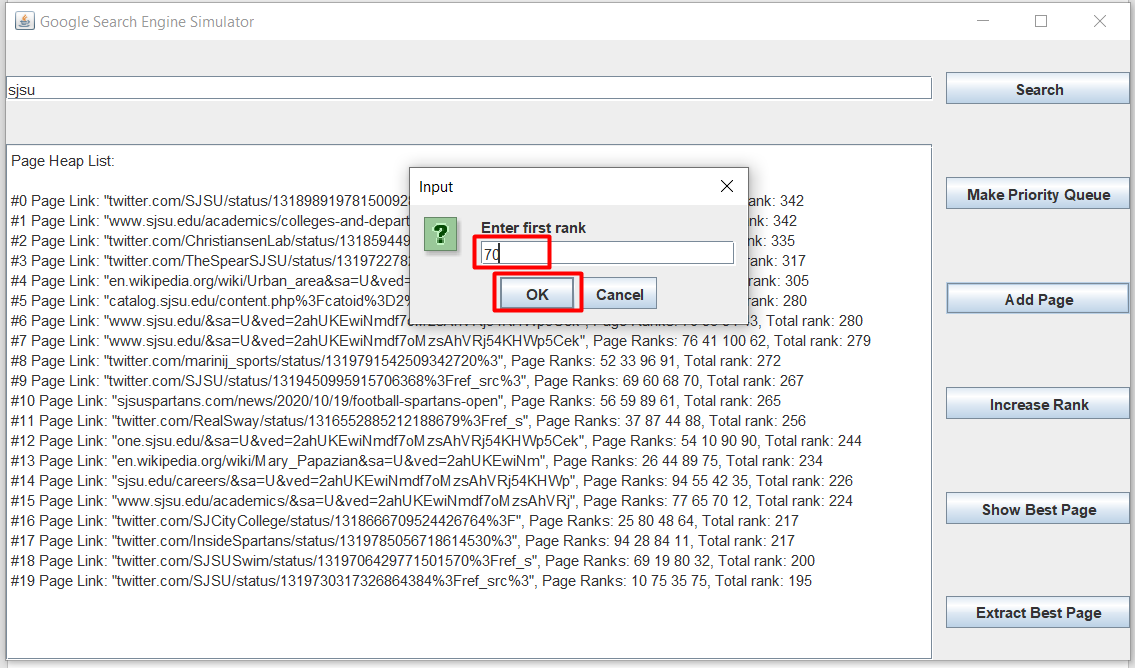




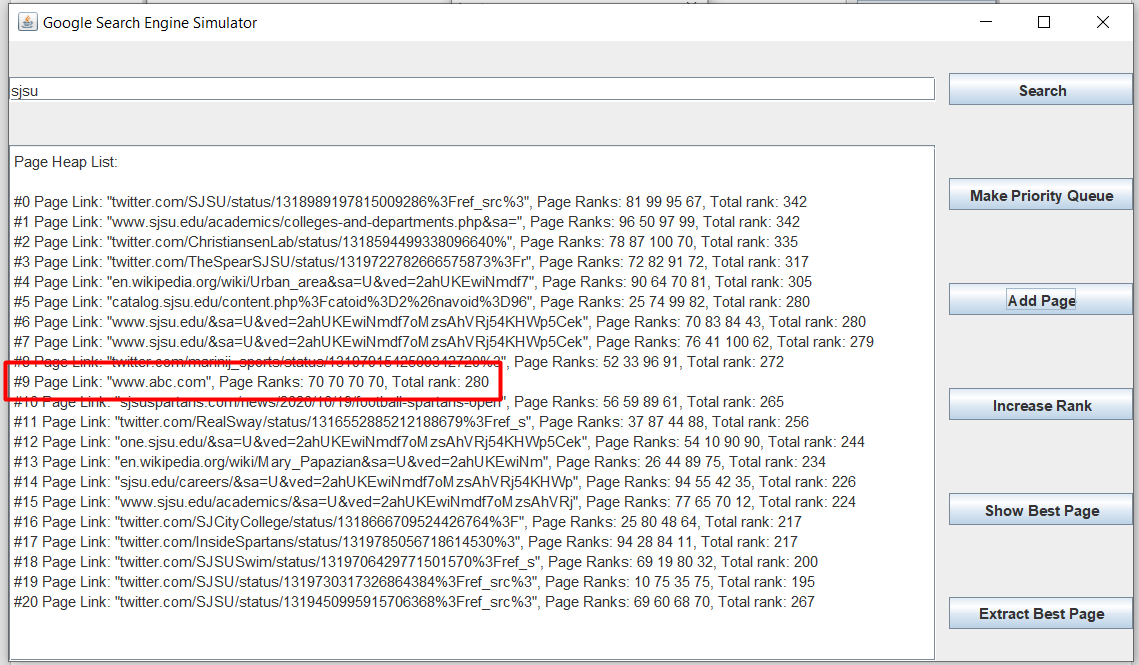
3)



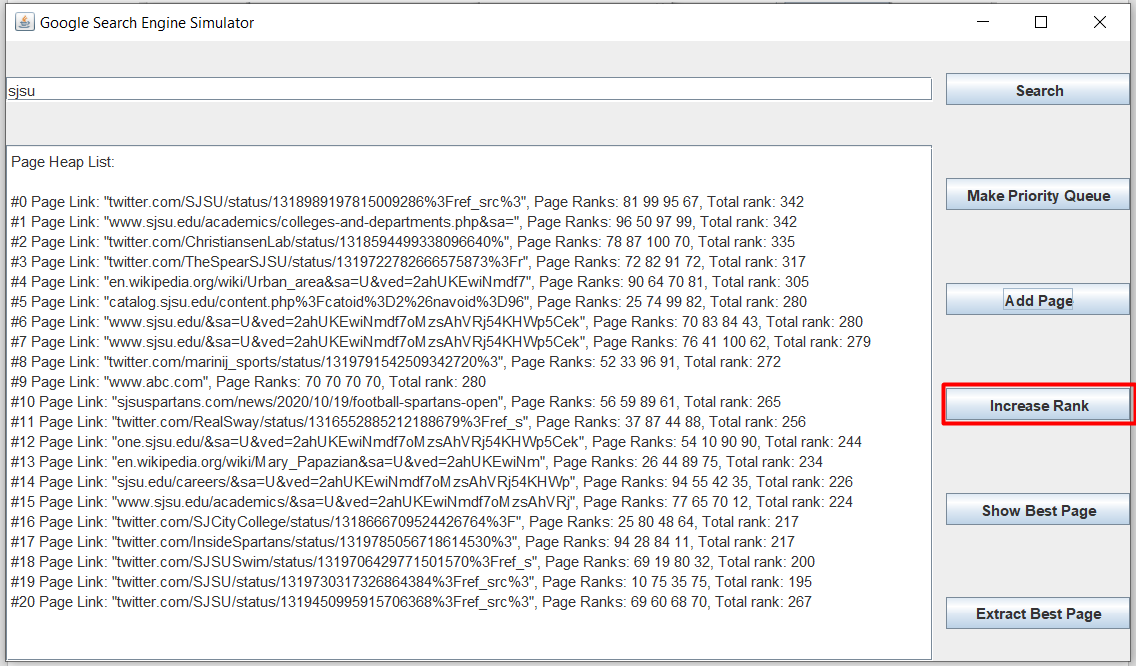


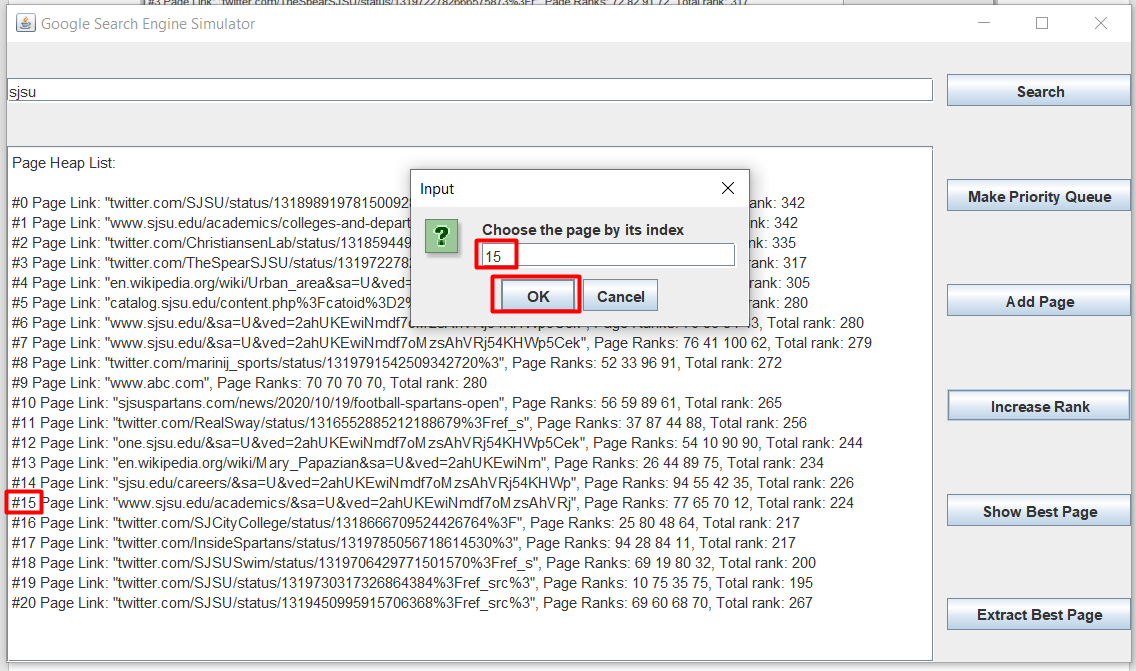


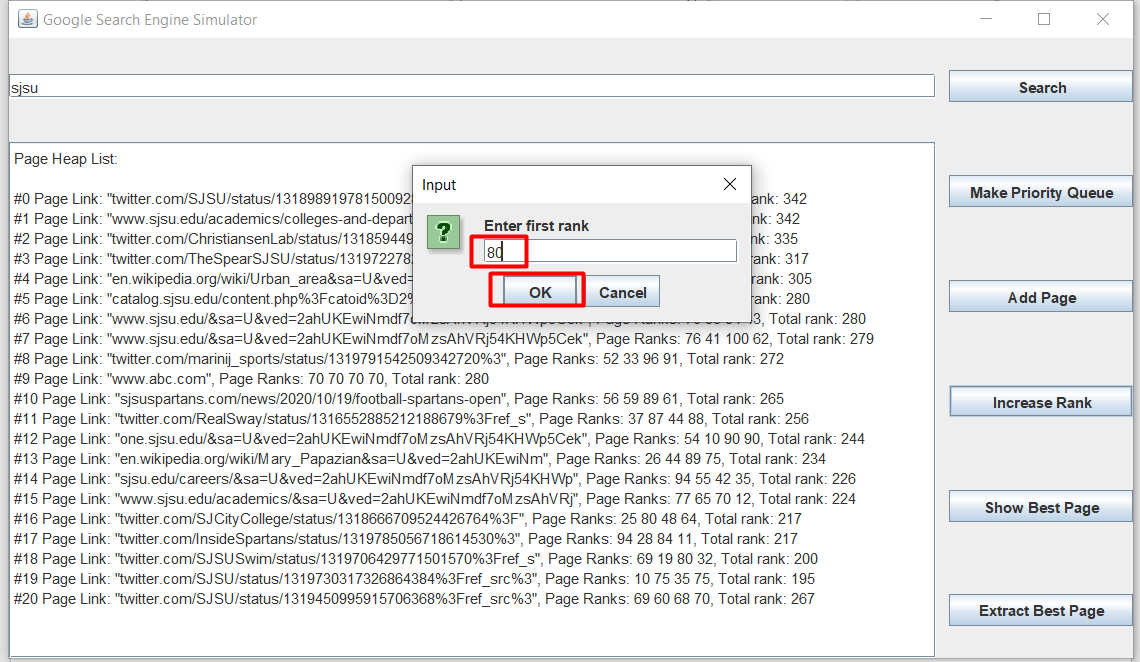
Repeat the same steps for the other 3 ranks



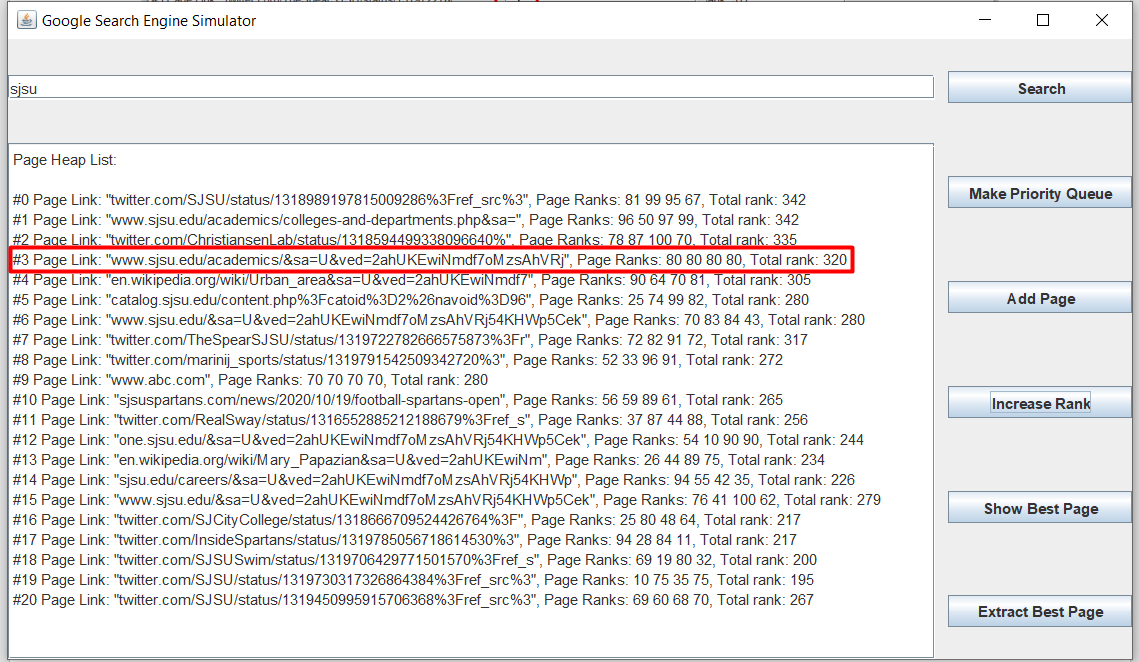
4)



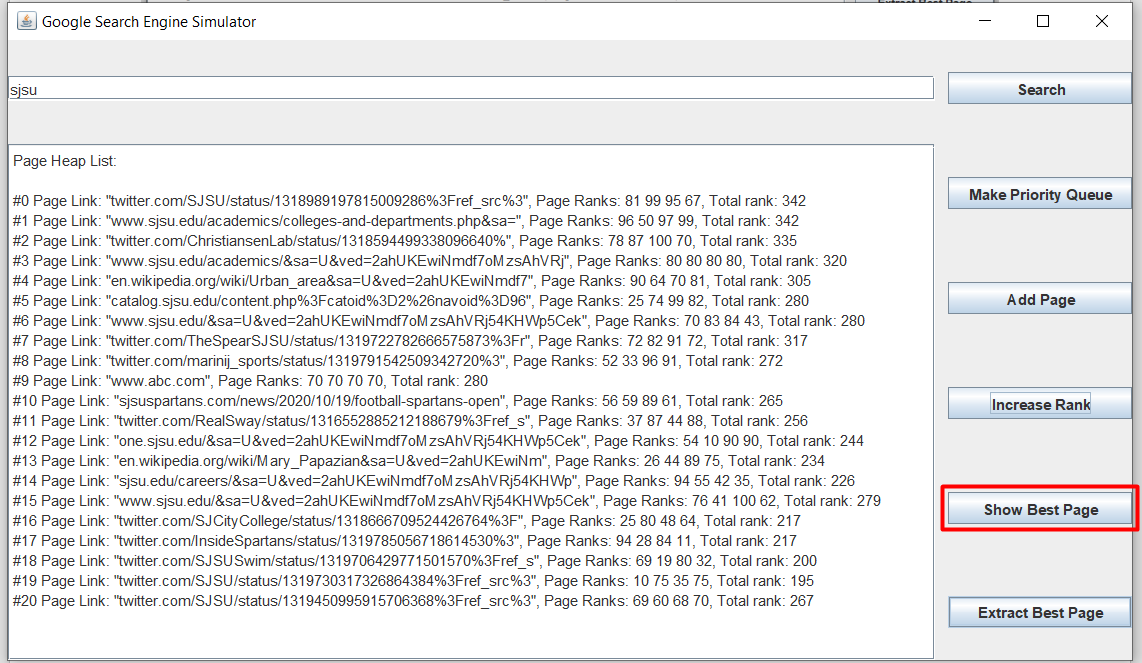


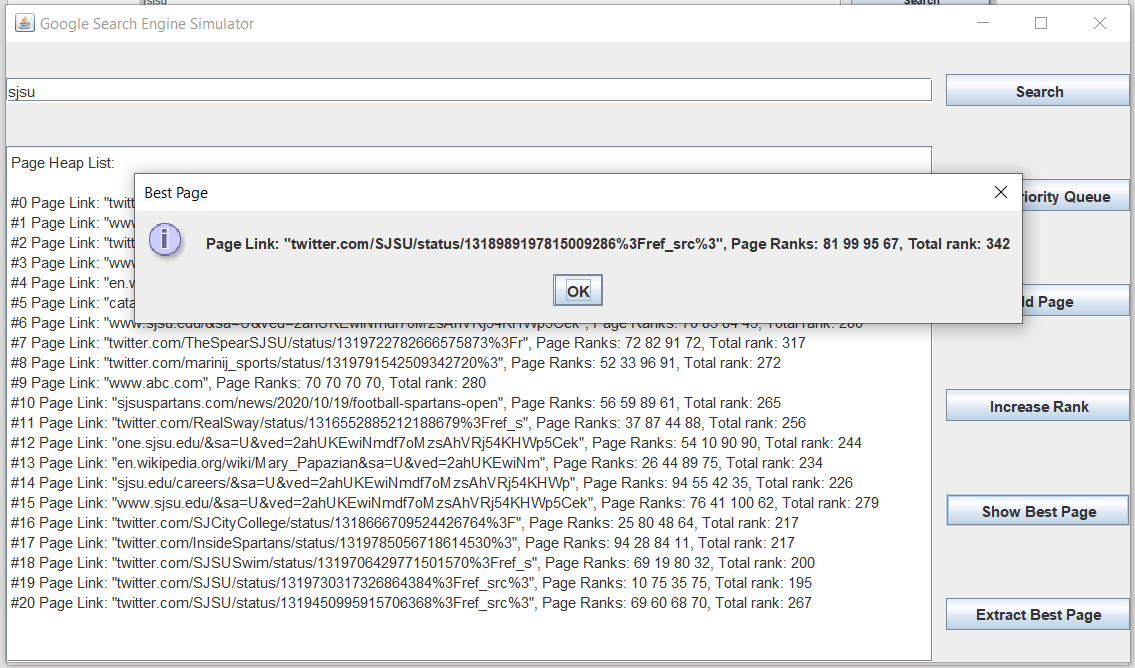


Repeat the same procedure for the other three ranks

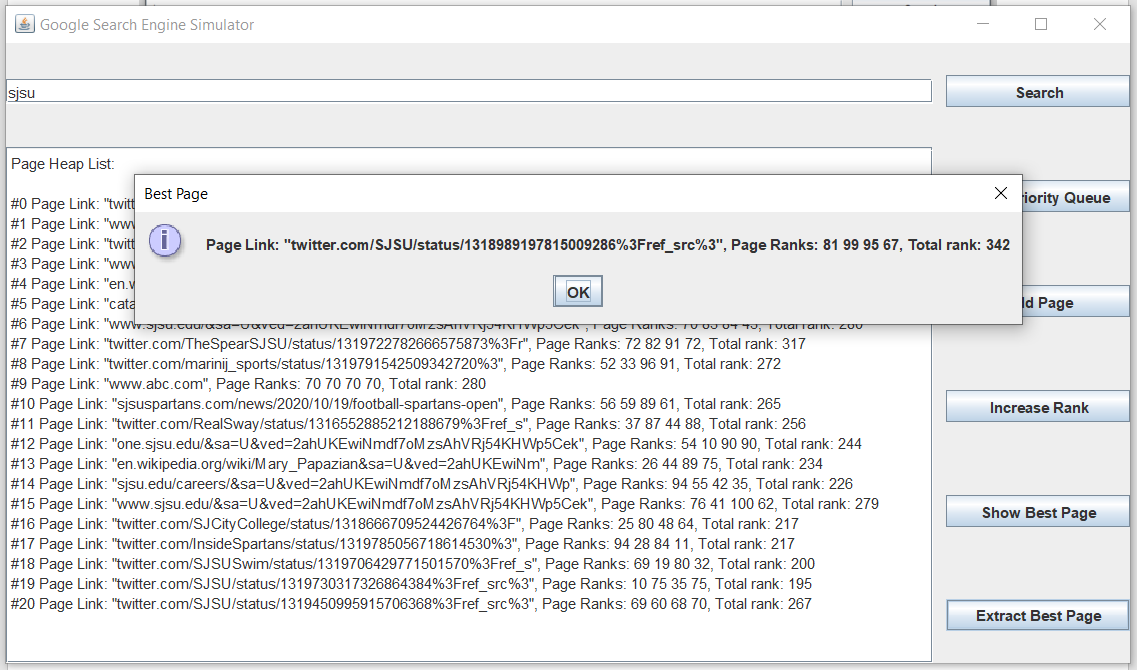


5)



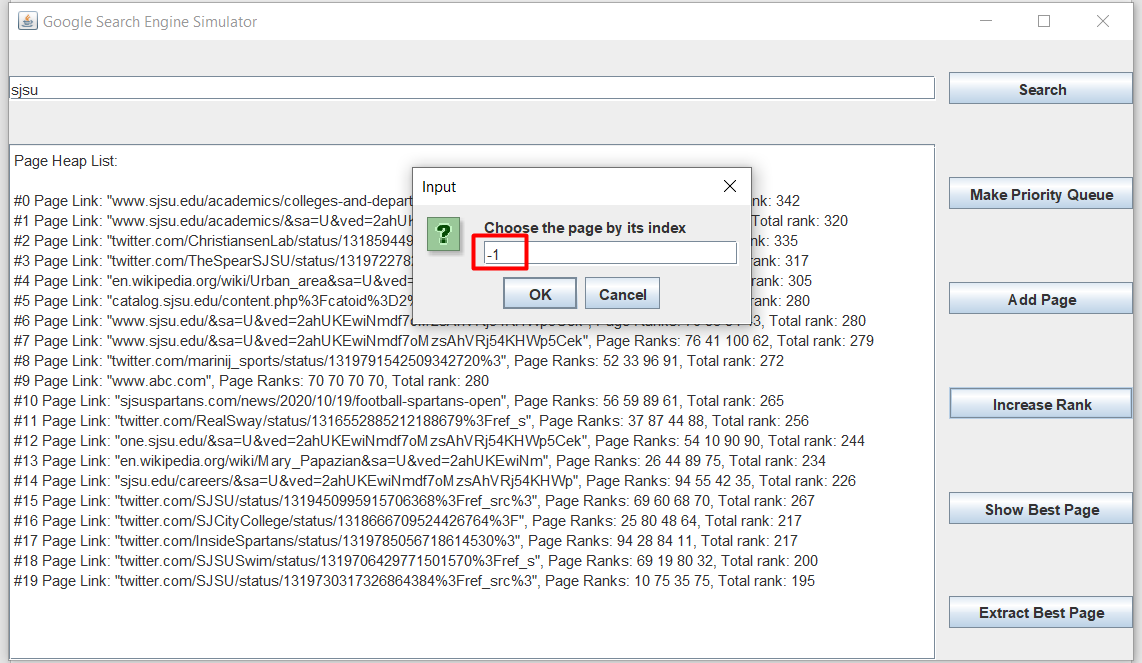


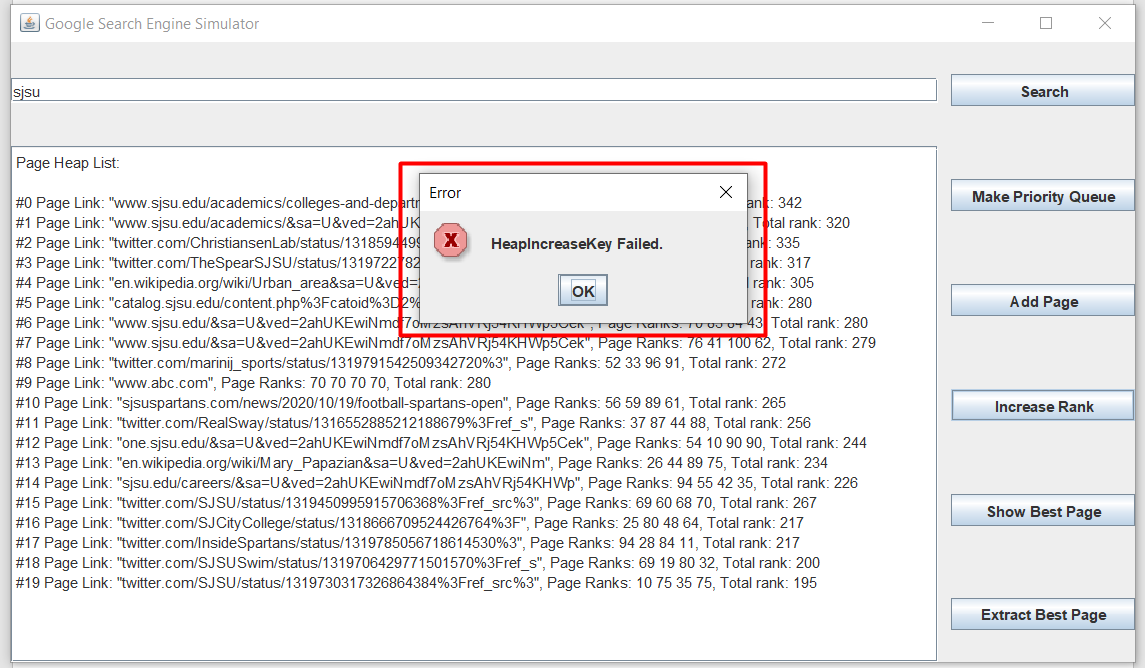






Exception handling (for example, in an increasing rank of the non-existing page):

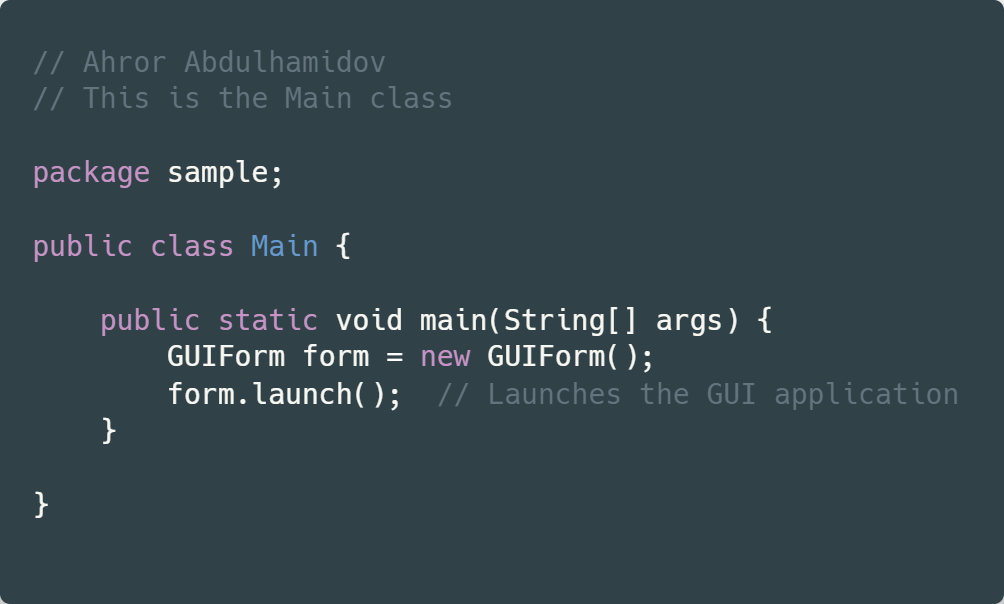




Other functionalities also have their own exception handlings, which look very similar to the one above.

**Source Code**

Main Class:



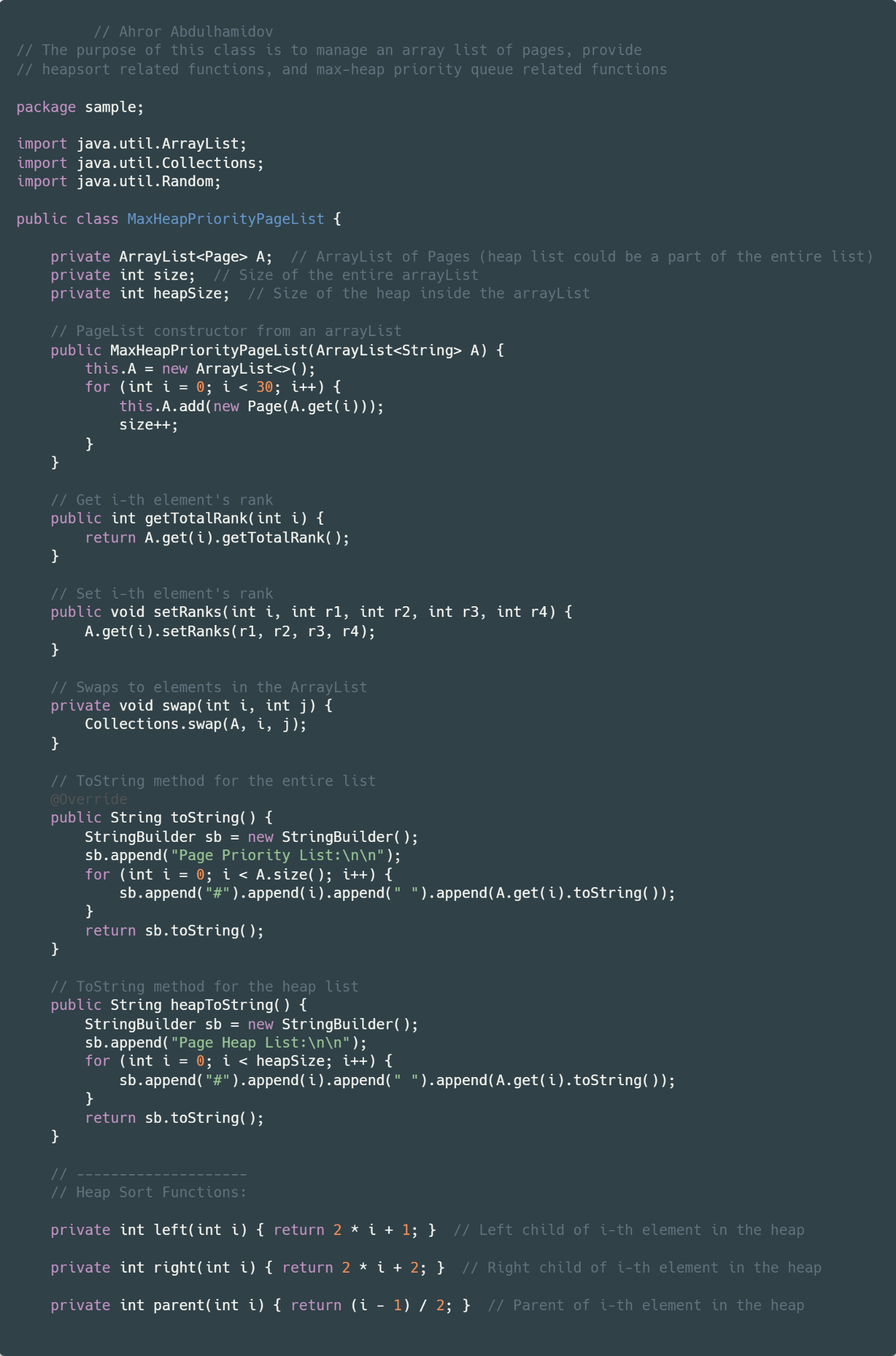
GUIForm Class:







MaxHeapPriorityPageList Class:







**Installation**

Simply unzip the PA1\_Ahror\_Abdulhamidov.zip file, and inside the folder, double click on the Search\_Engine\_Simulator.jar file, and the application window will show. All of the main features of this application are made sure to not crash, but rather, throw an error message dialog. The java SDK version used for this project is 11, and the IDE used is IntellIJ IDEA. If you have encountered any problem launching the application, try to install the right version of the Java SDK. For more information on the functionality, check the Front-end and Testing Screenshots sections of this document.

**Problems Encountered**

The main problem I encountered is the GUI implementation. At first, I learned how to make basic JavaFX applications, just so that I could implement it to this project, but I realized that it was nearly impossible to make an executable jar file (problems with sdk versions and etc.). I spent couple days on it, after which I decided to check Swing UI. With Swing UI, I realized that creating a jar file is very easy. I also spent couple hours when I was trying to create a table model view in my GUI, but configuring it seemed quite time consuming, so I disregarded it and went with using JTextPanel to views the Page list. And by the way, I realized that creating GUI is actually not that hard, especially with IntellIJ’s drag-and-drop form editor, which also took some time to figure out at first.

**Lessons Learned**

I learned several things while doing this project. First of all, it’s importance of choosing the right sorting algorithm and data structure. Here, we used heapsort and max-heap priority queue, because our data (Page list) has priorities, and having a max-heap is very helpful when using insert, extract, or key-increase methods (faster, lower running time). I also took a glance on to how a search engine works and how does the crawler work. I also learned a lot about making GUI with Java, and I regret that I didn’t learn it before I took my first data structure class in De Anza College.