Programming Assignment 1:
Google Search Engine Simulation

Yuxiao (Tom) Zheng
CS146-Section7

1. Unzip and Installation

2. Programming Design(Object-Oriented)

3. Implementation

4. classes/subroutines/function class:

5. Self-testing

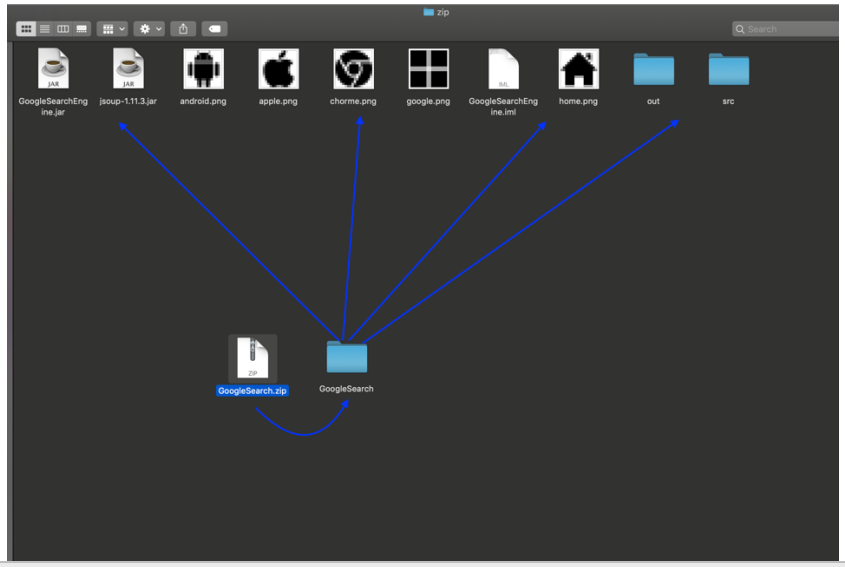
6. Problems encountered during implementation

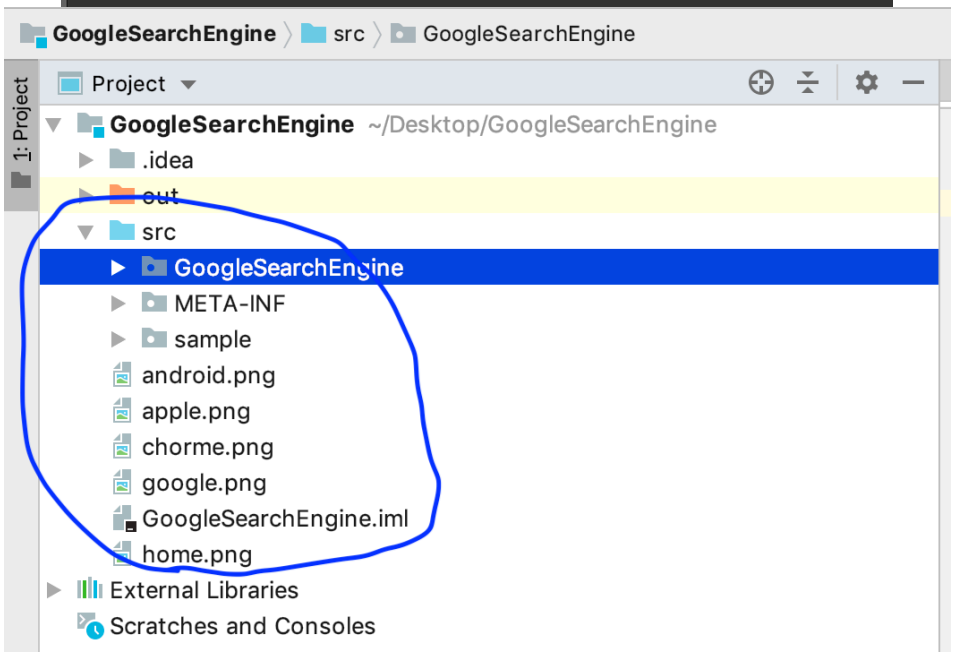
7. Leasson Learned

8. Updating (Debugging Report)

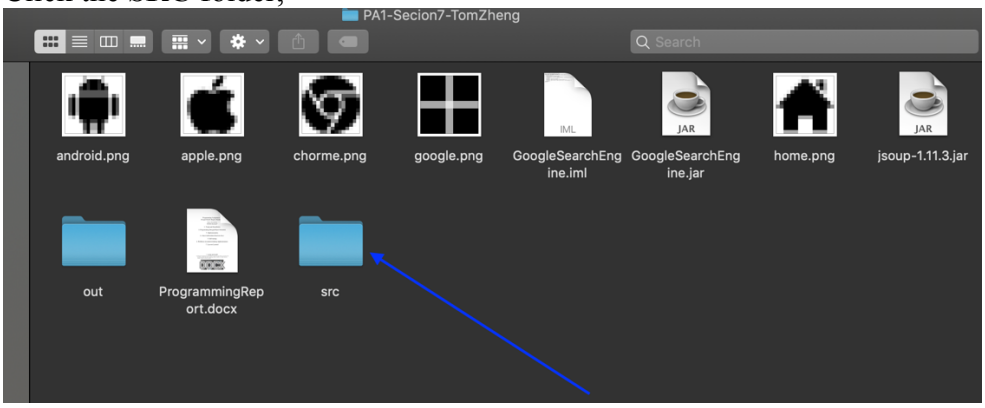
1. unzip and install

- Clicking the GoogleSearch.zip and unzip PA-1.zip into an empty folder, all .java files will be on the **src** folder. **Note**: the all .png imags must be outside of **src folders**, then the Intellij or Eclipse can find the .png file.

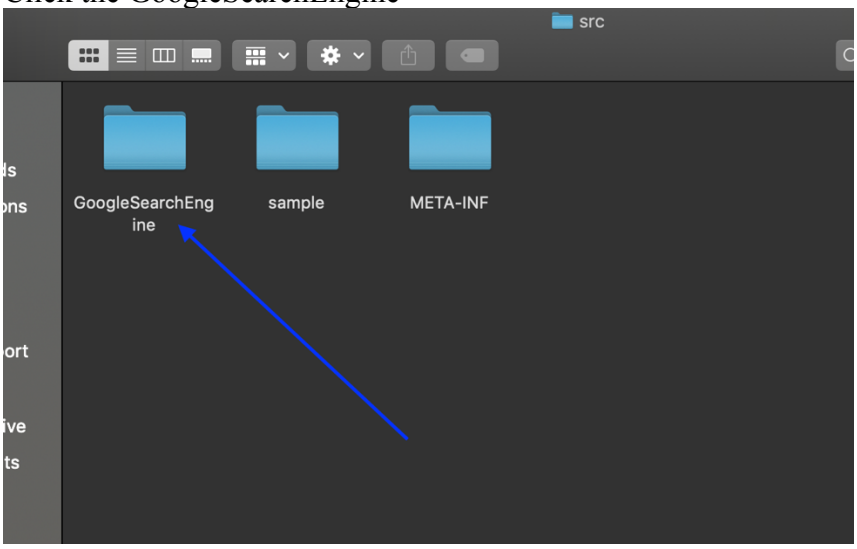


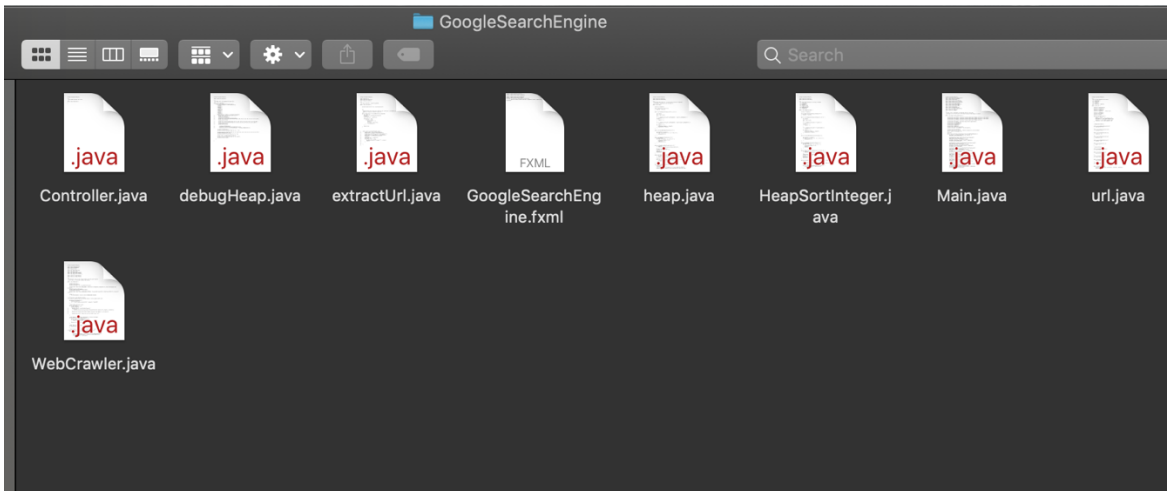


- Click the **SRC** folder,

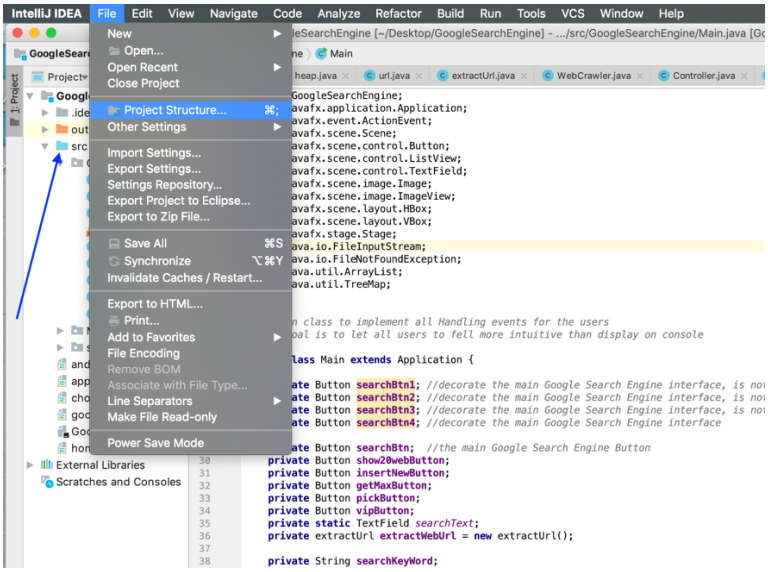


- Click the GoogleSearchEngine

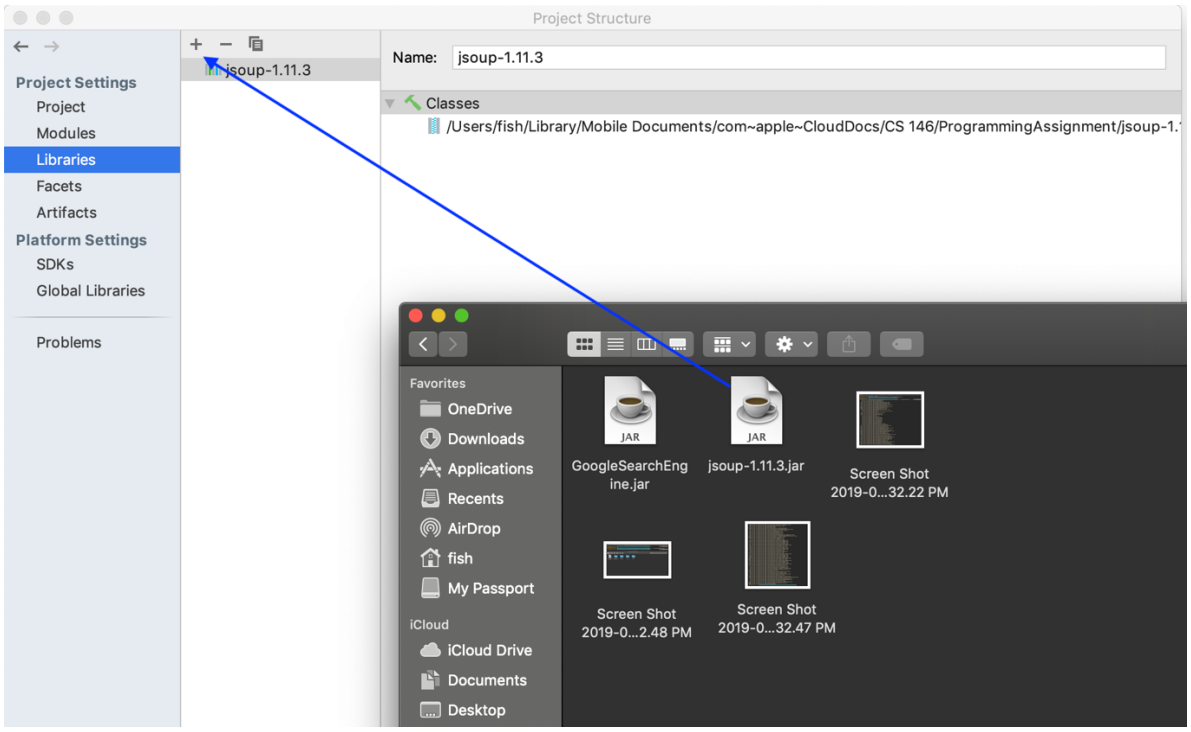




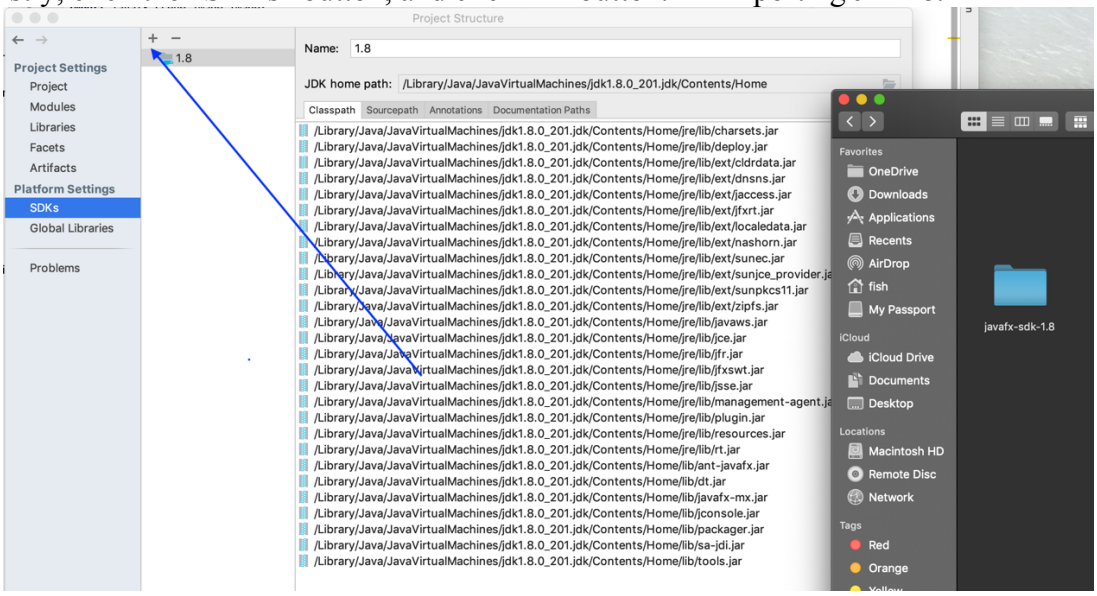
- Clicking the File→ click Project Structure. First creating a Package GoogleSearchEngine put all source files including .java files into the Package, this Package must be into **SRC** folder.



- After clicking Project Structure, click "+" it will import jsoup-1.11.3.jar. You will be using the all methods for extracting the web url by WebCrawler class.



- Note: Eclipse with JDK 8 under can export project into Runnable Jar, but not JDK 9. Therefore, I import JDK 8 for Intellij or Eclipse.
  Firstly, click the "SDKs" button, and click "+" button. → importing JDK 8.

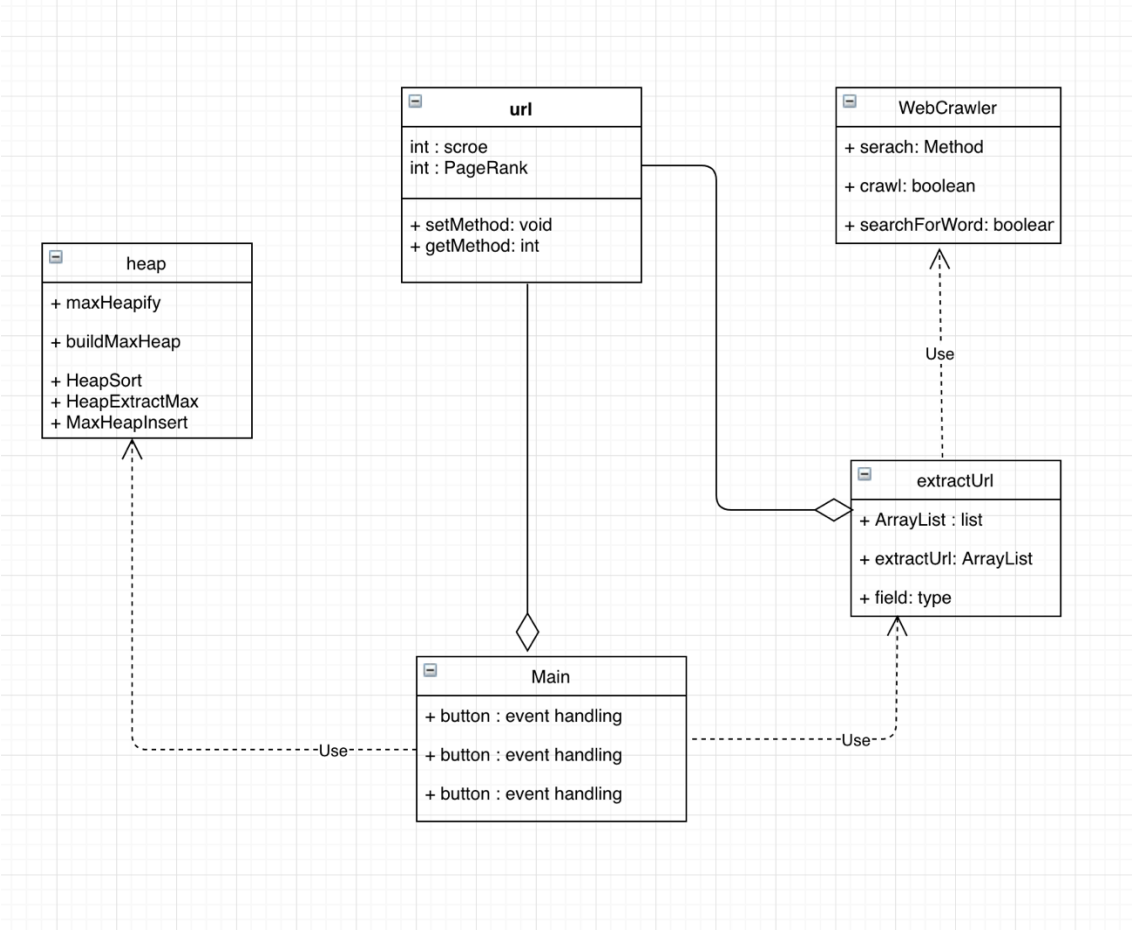


- To summary, all .java files → GoogleSearchEngine Package; all .png files is out of SRC folder; You must import JDK 8.

2. Programming Design

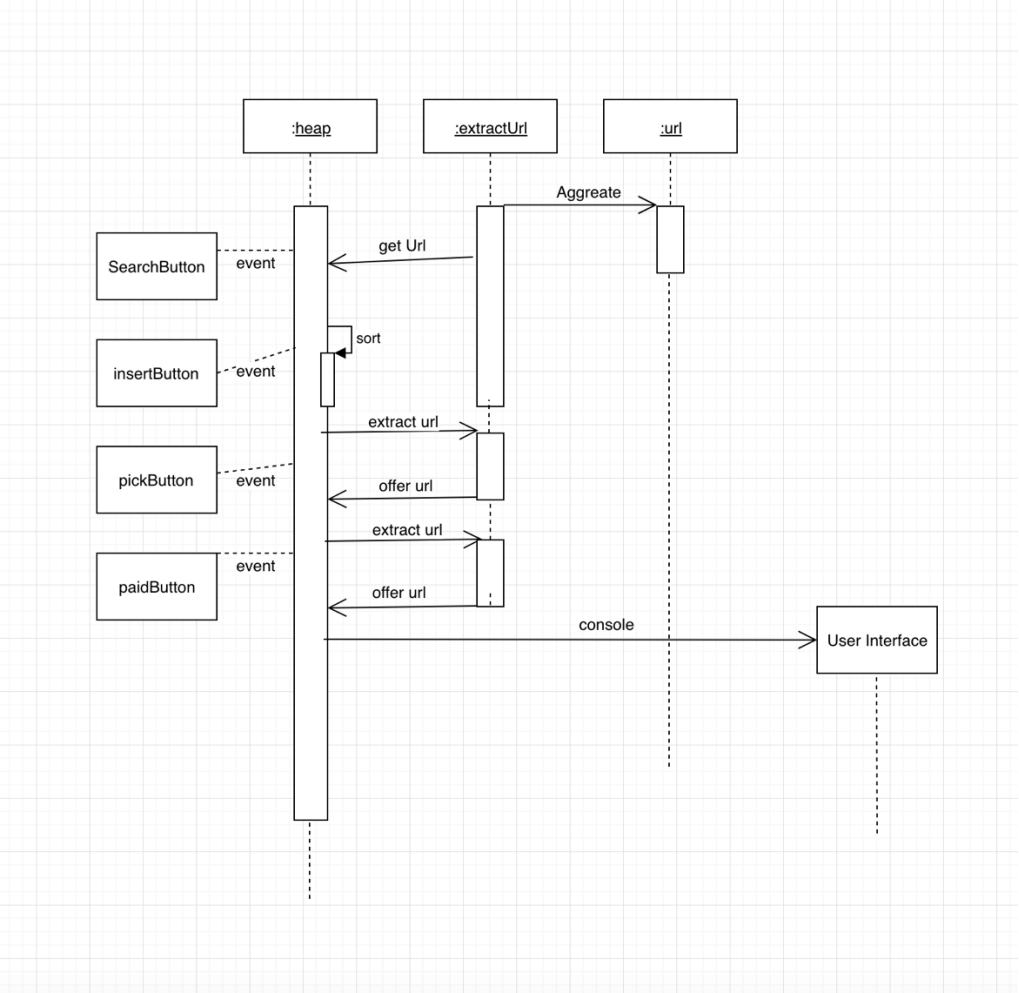
- Importing JavaFX lirbary, it will make the **USER INTERFACE** more percetual intuition, beautiful, and clean.
- Adding some .png icon images into this project, they will make the buttons more active and detailed.
- Importing the Jsoup lirbary into this project in order to use the WebCrawler class; we need WebCrawler to implement the extractUrl class.
- My extractUrl calss must aggreate the **URL** class, because all web urls that WebCrawler will capture, which will add into the ArrayList<url> for the Main class implementing different functioning for the users.
- The Main class depends on heap class and extractUrl, it was also aggreating with url class. The extractUrl class depends on WebCrawler class; it was also aggreating with url class.
- The different button-event method will using lambda expression to implement functioning.

There is a **UML diagram** below:



- The Main class will register any required event listeners on all the **Buttons** componets.

The bellow is the Main-class **Sequence Diagram**:

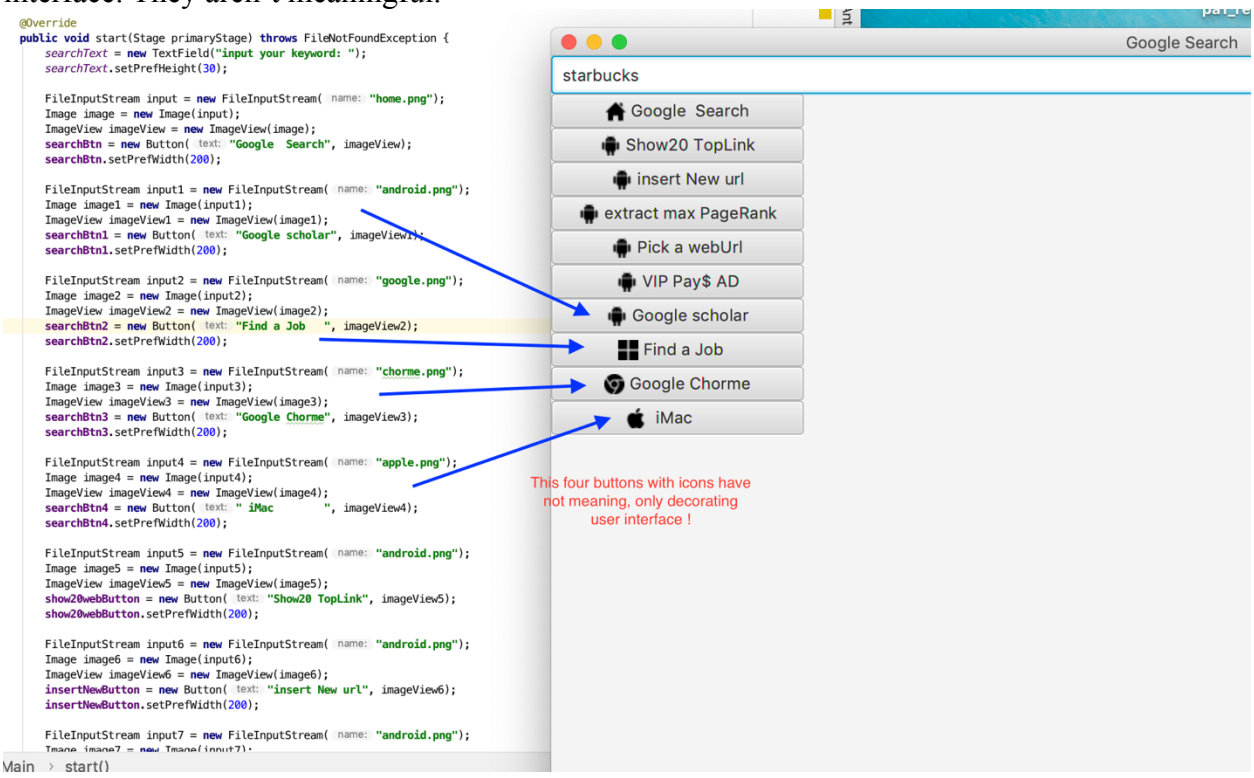


- All buttons will each fufill their purpose to finish different functioning excepct the **SearchButton** execute the primary task.
- The heap class will call iteself by invoking the method HeapSort in ascending order when the user hope to resort the ArrayList<url> to execute their appeals.
- In the lifetime, the extractUrl always play a large role to provide the web url that the users hope to get.

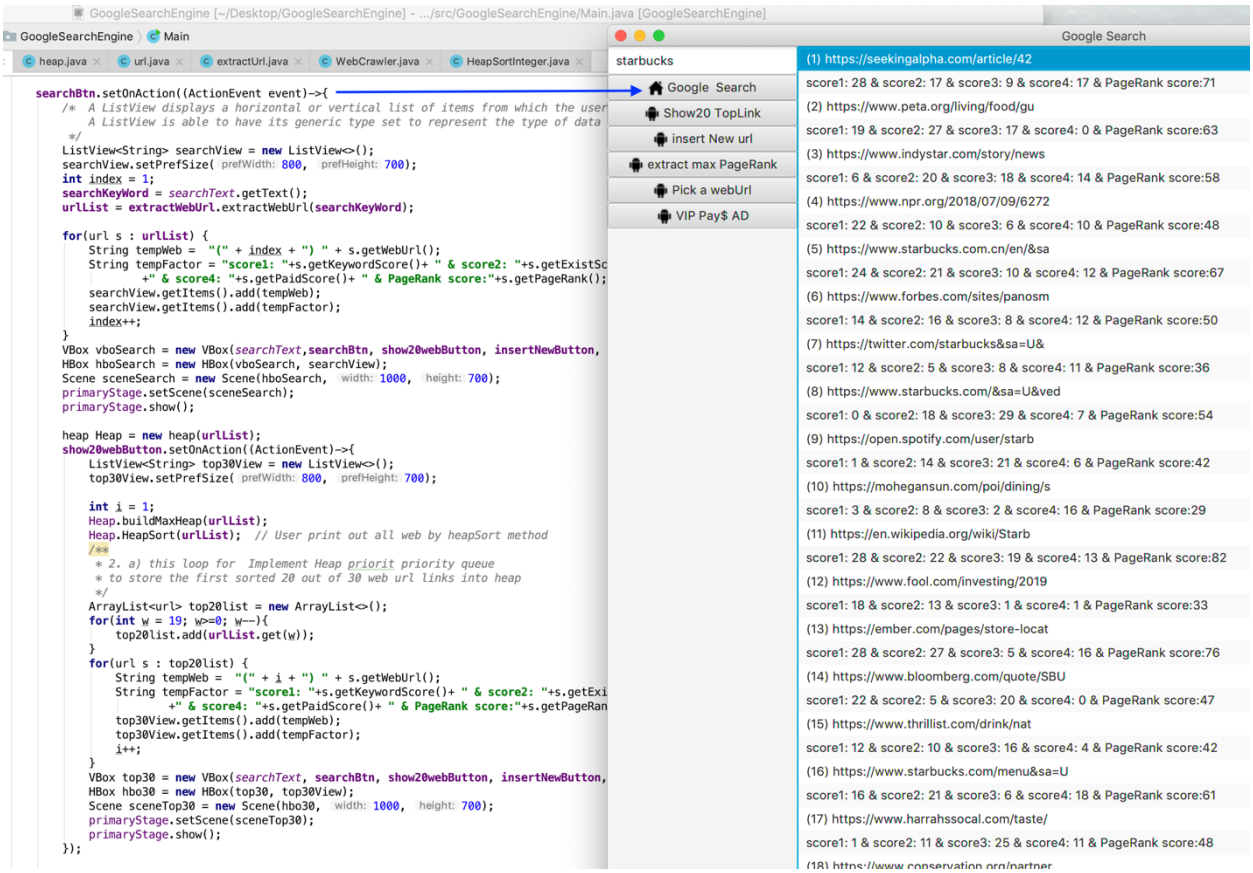
3. Implementing functioning

- The user UI Interface will hold the primary buttons: Google Search, Show20topLink, insertNewUrl, extract max PageRank, Pick a webUrl, and VIP Pay$ AD.

- Note: the Google scholar, Find a Job, Google Chorme, iMac is only to decorate the user interface. They aren't meaningful.

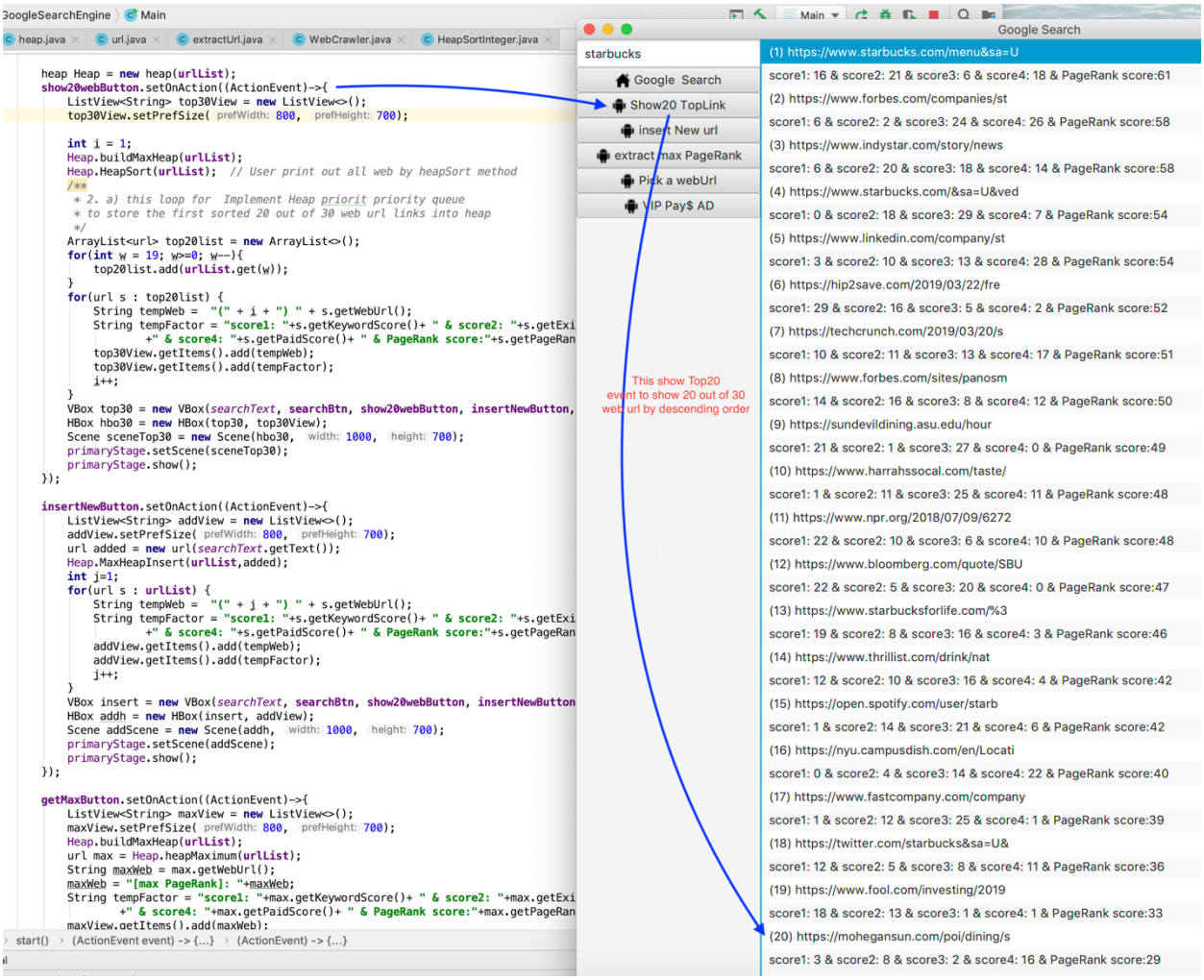


- When clicking the SearchButton, it will fire an event to all of the registered listeners with the object being updated, the cation of the update, the value before, and the value now.
- → calling the **extractWebUrl(KeyWord)** to get the web url into the ArrayList.
- → Printing out the first 30 elements.

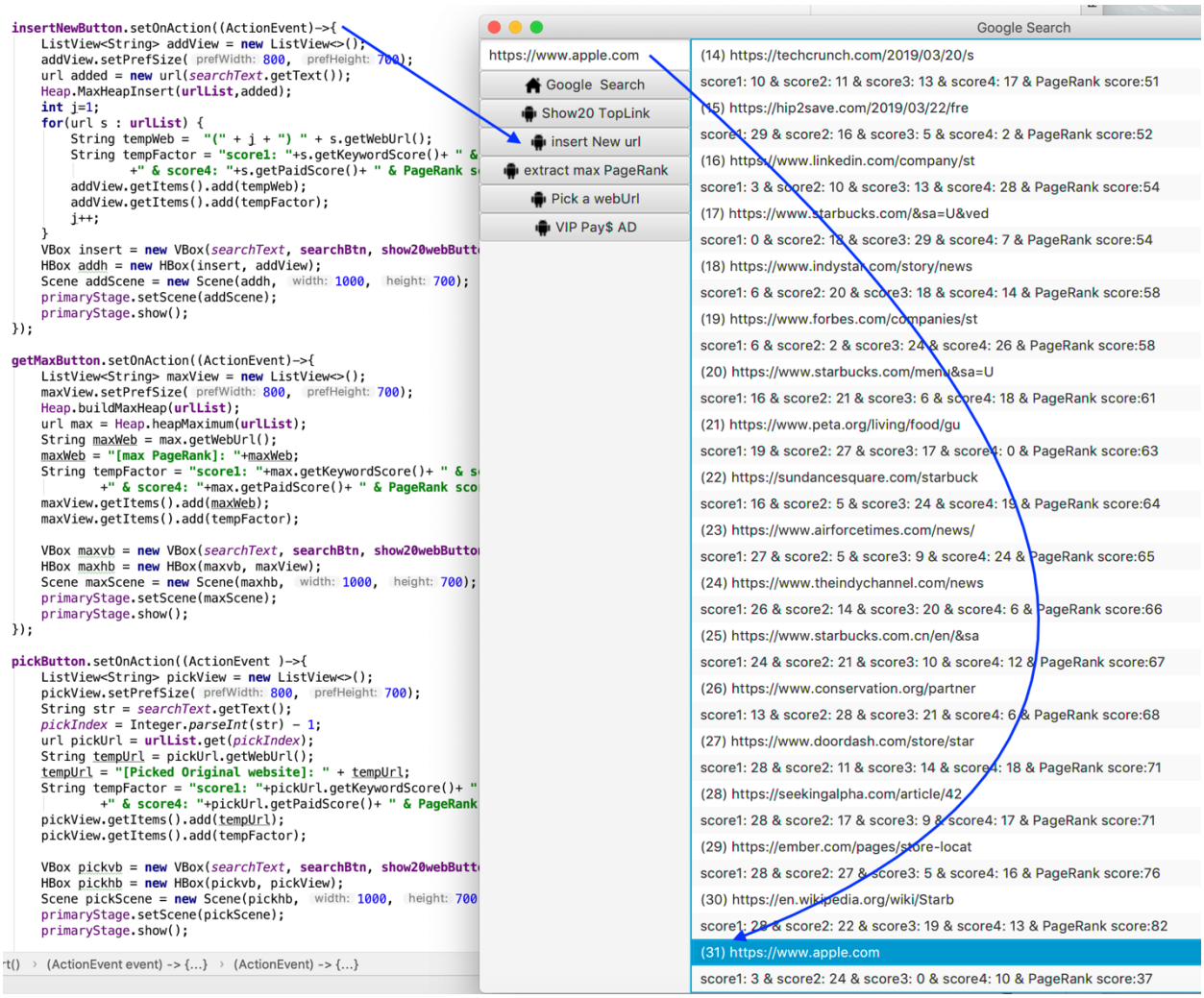


- Implementing 'Show20 TopLink' button → calling Heap.**buildMaxHeap(A)** and Heap.**HeapSort(A)**. Displaying the 20 out of 30 web url by descending order since the HeapSort method sorting the ArrayList in desceding order.

- The web url elments sorting like this: PageRank 61 → PageRank 58 → PageRank 54 → PageRank 52 → PageRank 51 → PageRank 50 … …

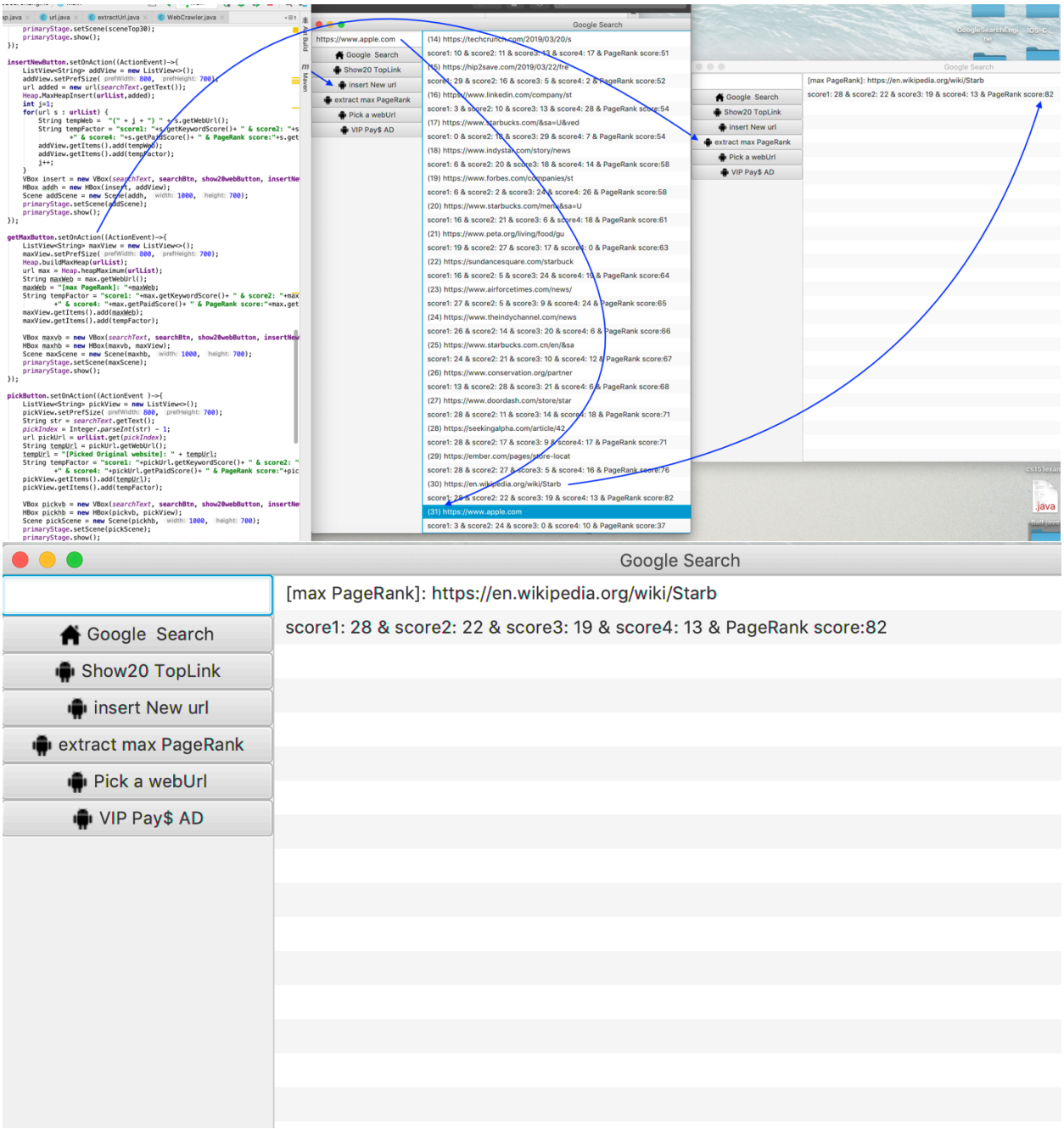


- Implementing “insert New url” button, insert Event → input https://www.apple.com → click **insert New url** Button → added a new web url into the heap By call MaxHeapInsert(A, key) mehod to implement the (31) element into the heap Tree !

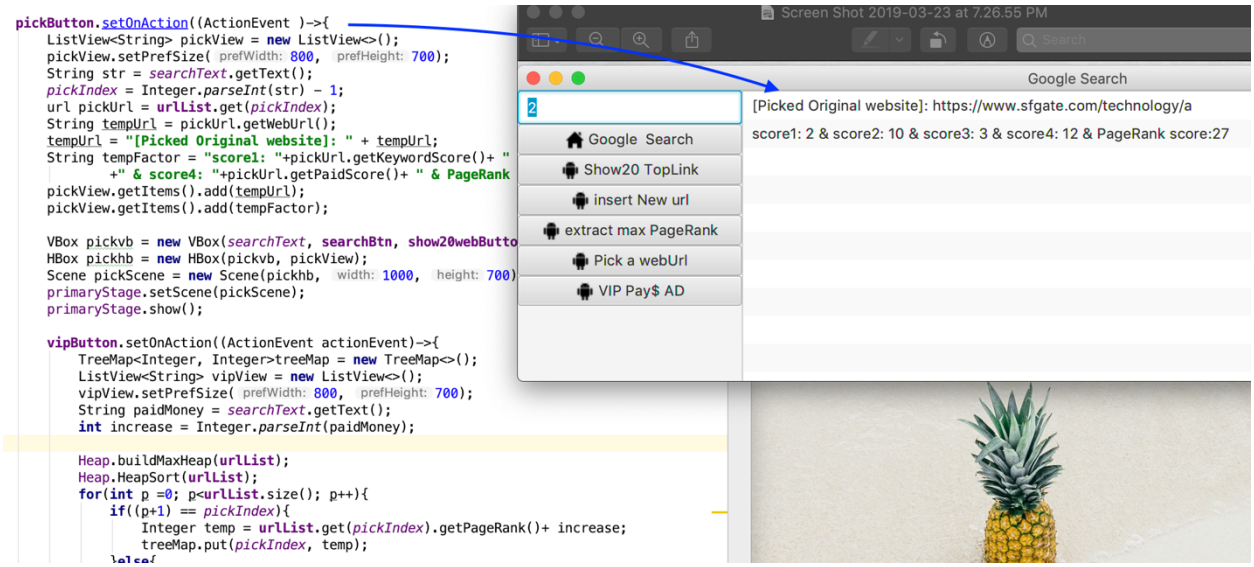


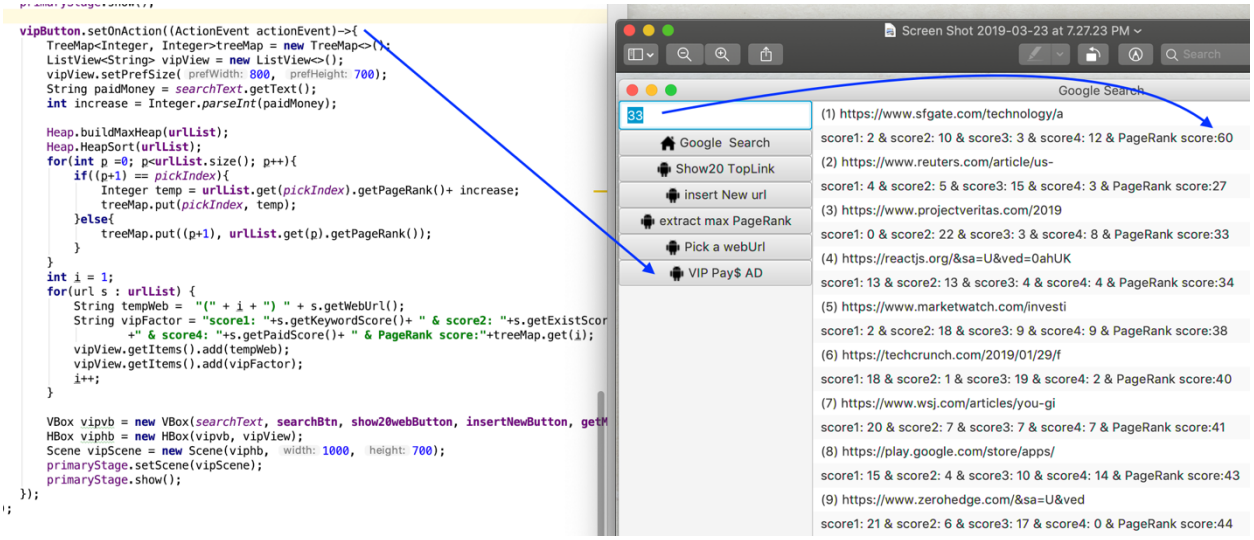
- Implementing “getMaxButton” → click button → calling Heap.**buildMaxHeap(A)**

→ calling Heap.heapMaximum(A) mehtod to get the max elment
→ displaying the result on the UI interface.



- Implementing pickButton and vipButton, → clicking "Pick a webUrl" → pick [www.sfgate.com] → clicking "VIP Pay$ AD" → add 33 rank score into original 27 rank → 33 + 27 = 60
- Including two steps: (1) → pick; (2) → pay $; (3) calculate total PageRank
- Create a new TreeMap → call method **TreeMap.put(key: Integer, Value: Integer)**;
  key using the index , value using the **getPageRank()**
  Finding the key → then finding the value → increasing the PageRank

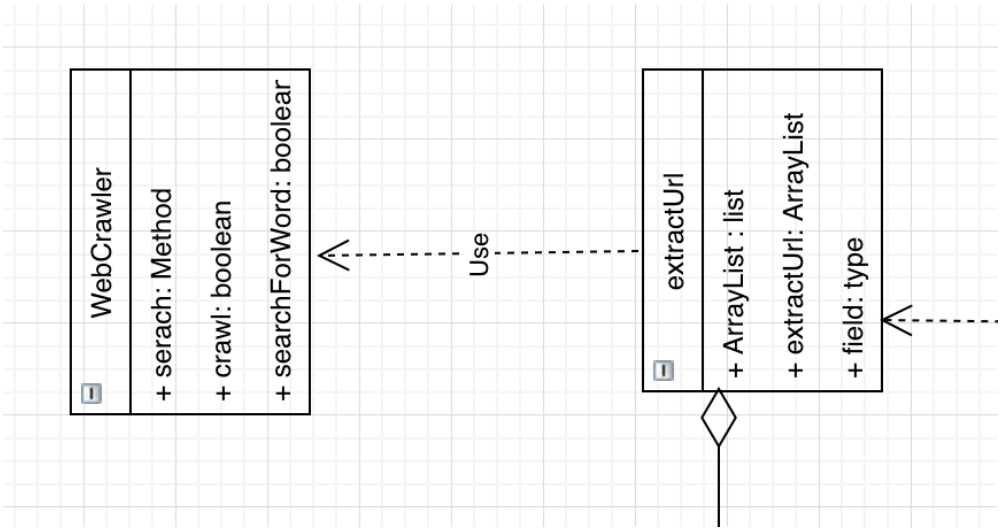




- To summary, so far so good! All functioning testing sucessfully. I have spent more than 120 hours!

4. classes/subroutines/function class:

- Class **extractUrl** → depending on Class **WebCrawler** :
  extractUrl's function **extractWebUrl(String keyword)** will help me get the website's address .
  constructor a new WebCrawler → calling wc.search()→ finding the required url elements → creating a new ArrayList<>() → A.dd(All weburls).




```
 */
public ArrayList<url> extractWebUrl(String keyword){
    int index = 1;
    WebCrawler c = new WebCrawler(keyword);
    c.search();
    Set<String> set = c.getUrls();

    for(String s : set){
        if(index <= 30){
            s = "https://" + s;
            index++;
            url weburl = new url(s);
            list.add(weburl);
        }
    }
    return list;
}
```

- Class **WebCrawler** has a constructor **WebCrawler(keyword)** to hold the keyword that the user inputed through Main class calling the **Handing Events**.
- It have imported the Jsoup lirbary, Jsoup lirbary is to help Crawler to get the url.

```java
import java.io.IOException;
import java.util.HashSet;
import java.util.Set;

import org.jsoup.Connection;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * A webCrawler class to get the web address with the given keywords
 * This class is to help get the URL, and testing
 */
public class WebCrawler {

    private String url;
    private String keyword;
    private Set<String> urls = new HashSet<>();

    private static final String USER_AGENT = "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)";
    private Document htmlDocument;
    private static Pattern patternDomainName;
    private Matcher matcher;
    private static final String DOMAIN_NAME_PATTERN = "([a-zA-Z0-9]([a-zA-Z0-9\\-]{0,61}[a-zA-Z0-9])?\\.)+[a-zA-Z]{2,6}";

    static {
        patternDomainName = Pattern.compile(DOMAIN_NAME_PATTERN);
    }
```

- The crawl(String url) method will crawl the links and put them in to a set to keep; Give it a URL and it makes an HTTP request for a web page.

```java
public boolean crawl(String url) {
    try {

        Connection connection = Jsoup.connect(url).userAgent(USER_AGENT);
        final Document htmlDocument = connection.timeout(5000).get();
        this.htmlDocument = htmlDocument;

//          if (connection.response().statusCode() == 200) {
//              System.out.println("\n**Visiting** Received web page at " + url);
//          }
        if (!connection.response().contentType().contains("text/html")) {
            System.out.println("**Failure** Retrieved something other than HTML");
            return false;
        }

        Elements linksOnPage = htmlDocument.select( cssQuery: "a[href]");
        System.out.println("Found (" + linksOnPage.size() + ") links");

        for (Element link : linksOnPage) {
            String temp = link.attr( attributeKey: "href");
            if (temp.startsWith("/url?q=http")) {
                this.urls.add(getDomainName(temp));
            }
        }
        return true;
    } catch (IOException ioe) {
        return false;
    }
}
```

- Heap class including seven method:
    1. maxHeapify → is a helper function to help buildMaxHeap to create a Heap Tree.

```java
public void maxHeapify(ArrayList<Integer> A, int i) {

    int largest;
    int l = 2 * i + 1;
    int r = 2 * i + 2;

    if (l < heapSize && A.get(l) > A.get(i)) {
        largest = l;
    }else{
        largest = i;
    }

    if (r < heapSize && A.get(r) > A.get(largest)) {
        largest = r;
    }
    if (largest != i) {
        Collections.swap(A, i, largest);
        maxHeapify(A, largest);
    }
}
```

2. buildMaxHeap → we can build a max-heap from an unordered array in linear time.

```java
public void buildMaxHeap(ArrayList<Integer> A) {

    this.heapSize = A.size();
    for (int i = (A.size() / 2) - 1; i >= 0; i--) {
        maxHeapify(A, i);
    }
}
```

3. HeapSort → After calling buildMaxHeap → sort an array in ascending order.

```java
public void HeapSort(ArrayList<Integer> A) {
    buildMaxHeap(A);
    for (int i = A.size() - 1; i > 0; i--) {
        Collections.swap(A, i, j: 0);
        heapSize--;
        maxHeapify(A, i: 0);
    }
}
```

4. HeapExtractMax → if heap.size < -1 throw new RuntimeExcption(“heap underflow”) → calling heapMaximum(A) → put the last element on the first position to replace the maximum → calling maxHeapify(A, 0) → return maximum

```
public url HeapExtractMax(ArrayList<url> A) {
    if (heapSize < 1) {
        throw new RuntimeException("heap underflow");
    }
    url max = heapMaximum(A);
    A.set(0, A.get(heapSize - 1));
    heapSize--;
    maxHeapify(A, i: 0);
    return max;
}
```

5. heapIncreaseKey → Increass the value of the element I's key to the new value k, where k>= I's current key value

```
public void heapIncreaseKey(ArrayList<url> A, int i, url key) {
    if (key.getPageRank() < A.get(i).getPageRank()) {
        throw new RuntimeException("new key is smaller than current key");
    }
    A.set(i, key);
    while (i > 0 && A.get(getParent(i)).getPageRank() < A.get(i).getPageRank()) {
        Collections.swap(A, i, getParent(i));
        i = getParent(i);
    }
}
```

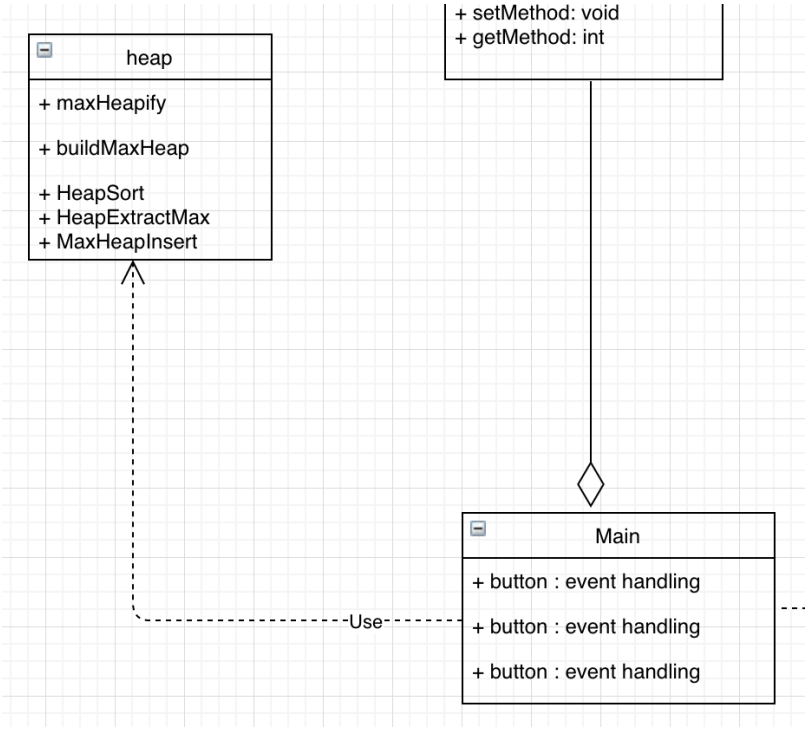
6. MaxHeapInsert → insert the key into the arraylist

```
public void MaxHeapInsert(ArrayList<url> A, url key) {
    heapSize++;
    A.add(key);
    A.get(heapSize - 1).setPageRank((int) Double.NEGATIVE_INFINITY);
    heapIncreaseKey(A, heapSize, key);//then give the leaf real value
}
```

7. heapMaximum → return the element of arraylist with the largest key

```
public url heapMaximum(ArrayList<url> A){
    return A.get(0);
}
```

- The Main class need Heap to implement different functioning : get Maxmimu, sort an array, in brief, Main depents on Heap.

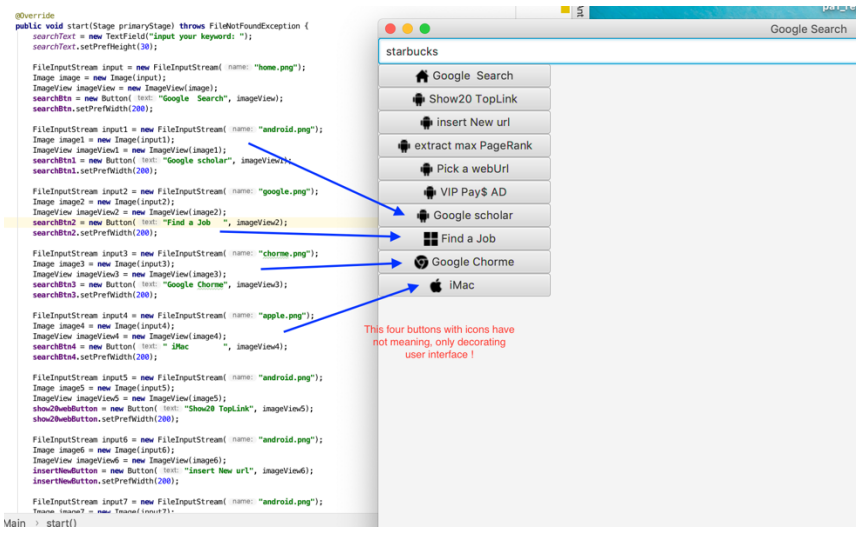


Main class method: start(Stage primaryStage)

- ListView → adding datas into the UI interface
- searchText.getText()→ save the data that the user input
- Vbox → vertical layout all buttons; the bottons will be vertical; contain the searchText
- Hbox → making vertical button on the left, the console of the UI interface will on the right
- Scene → contains the Vbox, Hbox, ListView
- primaryStage → holds all above components.

```
searchBtn.setOnAction((ActionEvent event)->{
    /*  A ListView displays a horizontal or vertical list of items from which the user may select
        A ListView is able to have its generic type set to represent the type of data in the backing
     */
    ListView<String> searchView = new ListView<>();
    searchView.setPrefSize( prefWidth: 800,  prefHeight: 700);
    int index = 1;
    searchKeyWord = searchText.getText();
    urlList = extractWebUrl.extractWebUrl(searchKeyWord);

    for(url s : urlList) {
        String tempWeb =  "(" + index + ") " + s.getWebUrl();
        String tempFactor = "score1: "+s.getKeywordScore()+ " & score2: "+s.getExistScore()+ " & scor
                +" & score4: "+s.getPaidScore()+ " & PageRank score:"+s.getPageRank();
        searchView.getItems().add(tempWeb);
        searchView.getItems().add(tempFactor);
        index++;
    }
    VBox vboSearch = new VBox(searchText,searchBtn, show20webButton, insertNewButton, getMaxButton, p
    HBox hboSearch = new HBox(vboSearch, searchView);
    Scene sceneSearch = new Scene(hboSearch,  width: 1000,  height: 700);
    primaryStage.setScene(sceneSearch);
    primaryStage.show();
```

- All buttons:



- Show20webButton : displays the top 20 web url in descending order

→ call HeapSort → arrayList adds all elments in descending order → print data

- ListView → adding datas into the UI interface
- searchText.getText()→ save the data that the user input
- Vbox → vertical layout all buttons; the bottons will be vertical; contain the searchText
- Hbox → making vertical button on the left, the console of the UI interface will on the right
- Scene → contains the Vbox, Hbox, ListView
- primaryStage → holds all above components.

- insertNewButton → add a new url elment
- getMaxButton → get the maximum url elent

- ListView → adding datas into the UI interface
- searchText.getText()→ save the data that the user input
- Vbox → vertical layout all buttons; the bottons will be vertical; contain the searchText
- Hbox → making vertical button on the left, the console of the UI interface will on the right
- Scene → contains the Vbox, Hbox, ListView
- primaryStage → holds all above components.

- pickButton → pick a required url elment
- vipButton → pay the money for google AD
- Note: vipButton inside the pickButton

- ListView → adding datas into the UI interface
- searchText.getText()→ save the data that the user input
- Vbox → vertical layout all buttons; the bottons will be vertical; contain the searchText
- Hbox → making vertical button on the left, the console of the UI interface will on the right
- Scene → contains the Vbox, Hbox, ListView
- primaryStage → holds all above components.

- Class url have a constructor: initialize score1 score2 score3 score 4

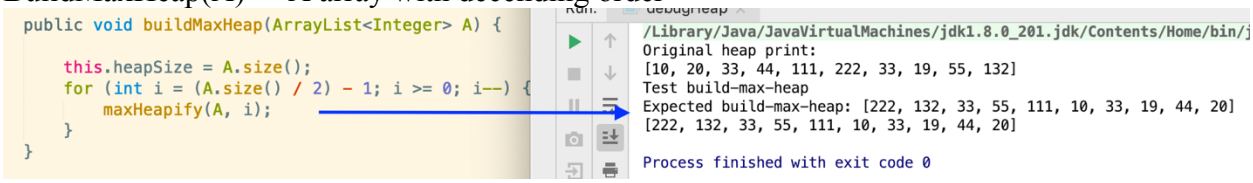
```
public url(String theUrl){
    webUrl = theUrl;
    paidScore = (int) (Math.random()* 30);
    keywordScore = (int) (Math.random()* 30);
    existScore = (int) (Math.random()* 30);
    linkScore = (int) (Math.random()* 30);
}
```

- Manipulate method && Asscess Method

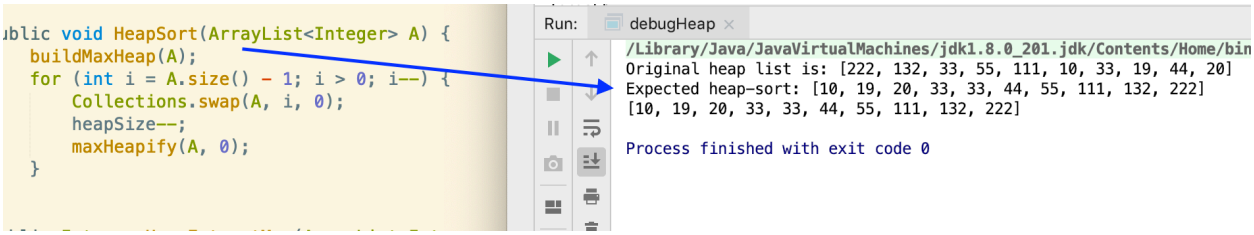
```
/* manipulate method */

public void setKeywordScore(int ks) { keywordScore = ks; }

public void setExistScore(int es) { existScore = es; }

public void setLinkScore(int ls) { linkScore = ls; }

public void setPaidScore(int ps) { paidScore = ps; }

public void setPageRank(int pr) { pageRank = pr; }

public void setIndex(int i) { index = i; }

/* Asscess Method*/

public int getKeywordScore() { return keywordScore; }

public int getExistScore() { return existScore; }

public int getLinkScore() { return linkScore; }

public int getPaidScore() { return paidScore; }

public int getIndex() { return index; }

public int getPageRank(){
    pageRank = keywordScore + existScore + linkScore + paidScore;
    return pageRank;
}

public String getWebUrl() { return webUrl; }
```

5. Self-tesing screen shots

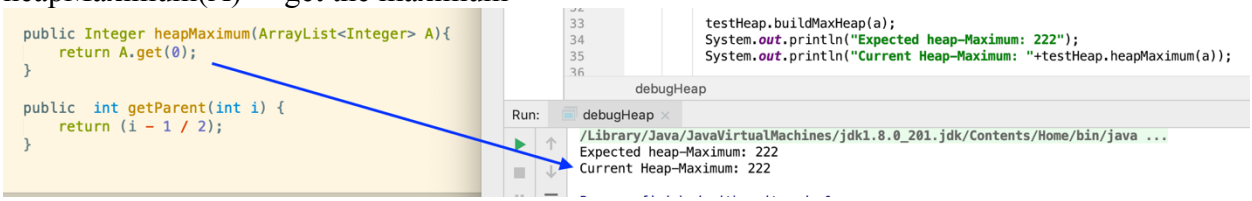
- BuildMaxHeap(A) → A array with decending order



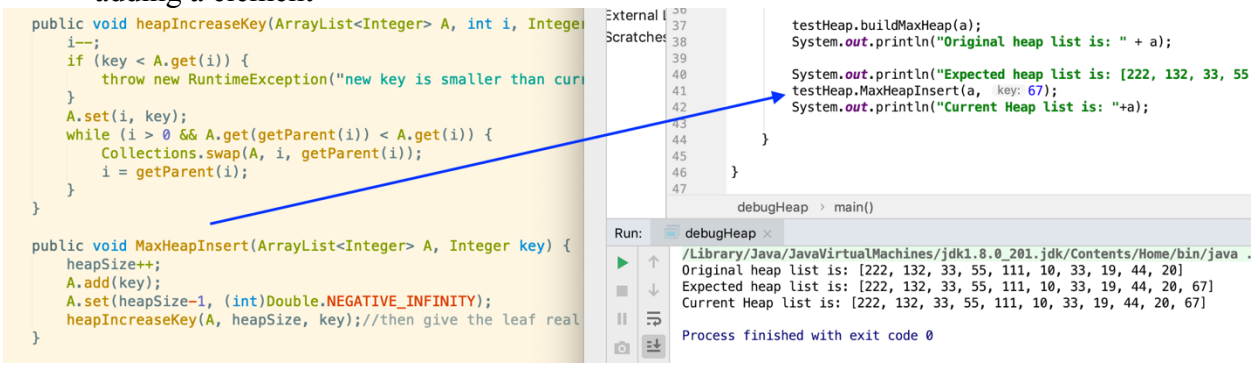
- HeapSort(A) → An array sorts with acending order



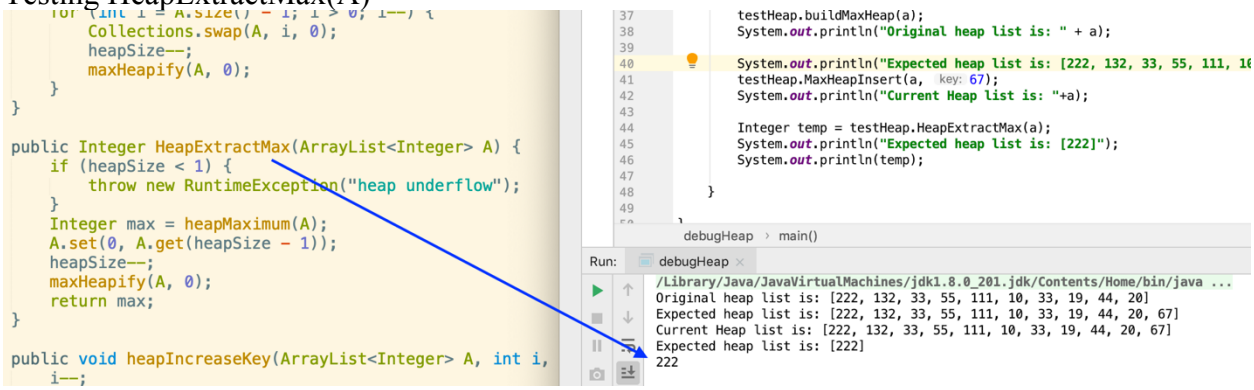
- heapMaximum(A) → get the maximum



- tesing heapIncresekey() && MaxHeapInsert()
  → adding a element



- Testing HeapExtractMax(A)



```
        Collections.swap(A, i, 0);
        heapSize--;
        maxHeapify(A, 0);
    }
}

public Integer HeapExtractMax(ArrayList<Integer> A) {
    if (heapSize < 1) {
        throw new RuntimeException("heap underflow");
    }
    Integer max = heapMaximum(A);
    A.set(0, A.get(heapSize - 1));
    heapSize--;
    maxHeapify(A, 0);
    return max;
}

public void heapIncreaseKey(ArrayList<Integer> A, int i,
    i--;
```

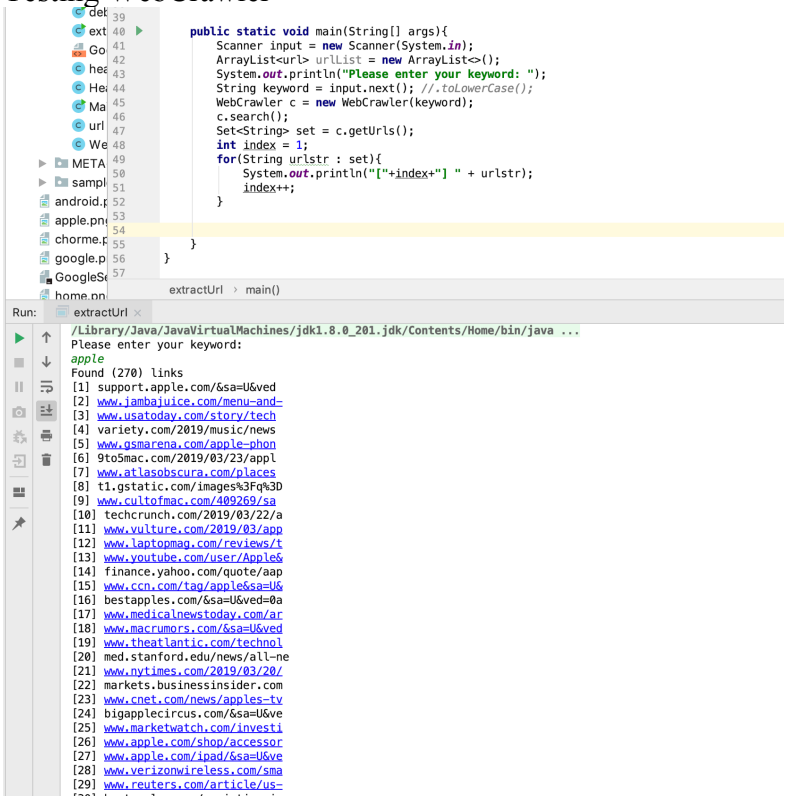
```
        testHeap.buildMaxHeap(a);
        System.out.println("Original heap list is: " + a);

        System.out.println("Expected heap list is: [222, 132, 33, 55, 111, 10
        testHeap.MaxHeapInsert(a,  key: 67);
        System.out.println("Current Heap list is: "+a);

        Integer temp = testHeap.HeapExtractMax(a);
        System.out.println("Expected heap list is: [222]");
        System.out.println(temp);

    }
}
```

debugHeap › main()

Run: debugHeap

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java ...
Original heap list is: [222, 132, 33, 55, 111, 10, 33, 19, 44, 20]
Expected heap list is: [222, 132, 33, 55, 111, 10, 33, 19, 44, 20, 67]
Current Heap list is: [222, 132, 33, 55, 111, 10, 33, 19, 44, 20, 67]
Expected heap list is: [222]
222
```

- Testing WebCrawler



```
    public static void main(String[] args){
        Scanner input = new Scanner(System.in);
        ArrayList<url> urlList = new ArrayList<>();
        System.out.println("Please enter your keyword: ");
        String keyword = input.next(); //.toLowerCase();
        WebCrawler c = new WebCrawler(keyword);
        c.search();
        Set<String> set = c.getUrls();
        int index = 1;
        for(String urlstr : set){
            System.out.println("["+index+"] " + urlstr);
            index++;
        }

    }
}
```

extractUrl › main()

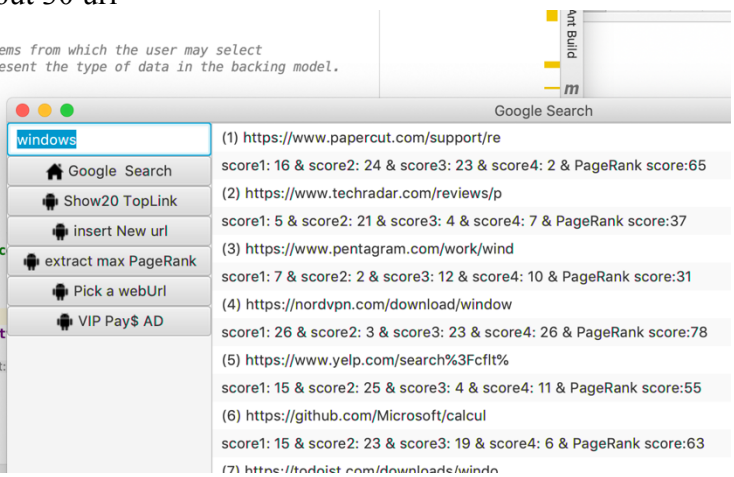
Run: extractUrl

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java ...
Please enter your keyword:
apple
Found (270) links
[1] support.apple.com/&sa=U&ved
[2] www.jambajuice.com/menu-and-
[3] www.usatoday.com/story/tech
[4] variety.com/2019/music/news
[5] www.gsmarena.com/apple-phon
[6] 9to5mac.com/2019/03/23/appl
[7] www.atlasobscura.com/places
[8] t1.gstatic.com/images%3Fq%3D
[9] www.cultofmac.com/409269/sa
[10] techcrunch.com/2019/03/22/a
[11] www.vulture.com/2019/03/app
[12] www.laptopmag.com/reviews/t
[13] www.youtube.com/user/Apple&
[14] finance.yahoo.com/quote/aap
[15] www.ccn.com/tag/apple&sa=U&
[16] bestapples.com/&sa=U&ved=0a
[17] www.medicalnewstoday.com/ar
[18] www.macrumors.com/&sa=U&ved
[19] www.theatlantic.com/technol
[20] med.stanford.edu/news/all-ne
[21] www.nytimes.com/2019/03/20/
[22] markets.businessinsider.com
[23] www.cnet.com/news/apples-tv
[24] bigapplecircus.com/&sa=U&ve
[25] www.marketwatch.com/investi
[26] www.apple.com/shop/accessor
[27] www.apple.com/ipad/&sa=U&ve
[28] www.verizonwireless.com/sma
[29] www.reuters.com/article/us-
```

-

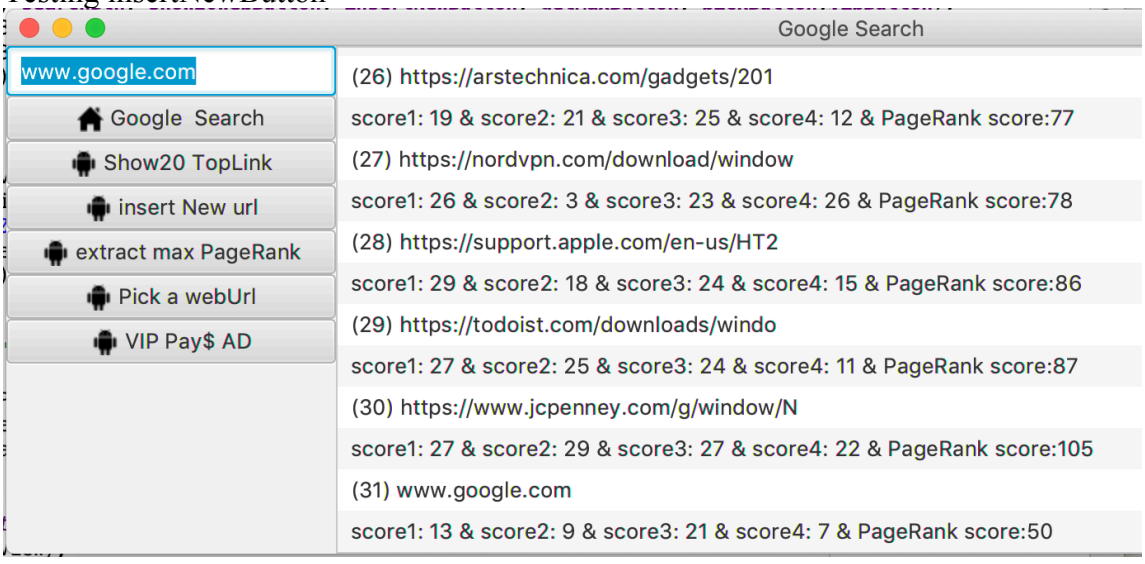
- Testing searchBtn → input “windows” out 30 url

```
searchBtn.setOnAction((ActionEvent event)->{
    /*  A ListView displays a horizontal or vertical list of items from which the user may select
        A ListView is able to have its generic type set to represent the type of data in the backing model.
     */
    ListView<String> searchView = new ListView<>();
    searchView.setPrefSize( prefWidth: 800,  prefHeight: 700);
    int index = 1;
    searchKeyWord = searchText.getText();
    urlList = extractWebUrl.extractWebUrl(searchKeyWord);

    for(url s : urlList) {
        String tempWeb =  "(" + index + ") " + s.getWebUrl();
        String tempFactor = "score1: "+s.getKeywordScore()+ " &
                +" & score4: "+s.getPaidScore()+ " & PageRank sc
        searchView.getItems().add(tempWeb);
        searchView.getItems().add(tempFactor);
        index++;
    }
    VBox vboSearch = new VBox(searchText,searchBtn, show20webBut
    HBox hboSearch = new HBox(vboSearch, searchView);
    Scene sceneSearch = new Scene(hboSearch,  width: 1000,  height:
    primaryStage.setScene(sceneSearch);
    primaryStage.show();

    heap Heap = new heap(urlList);
    show20webButton.setOnAction((ActionEvent)->{
```

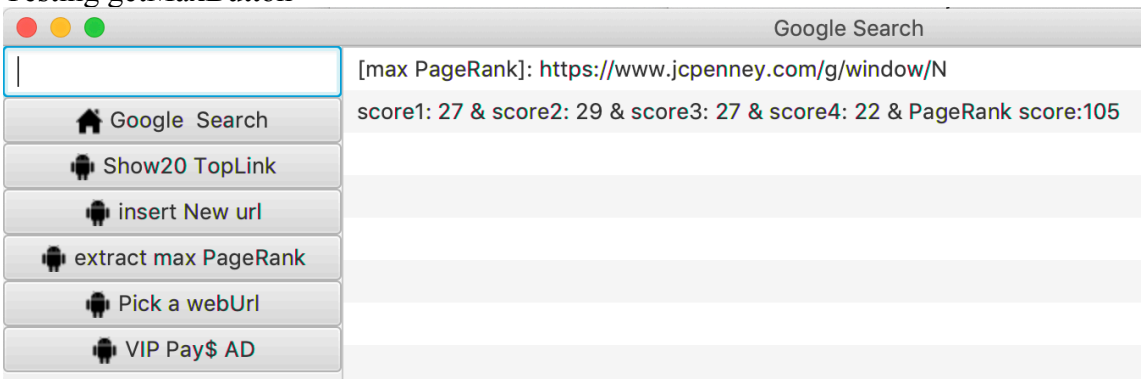


- Testing show20webButton

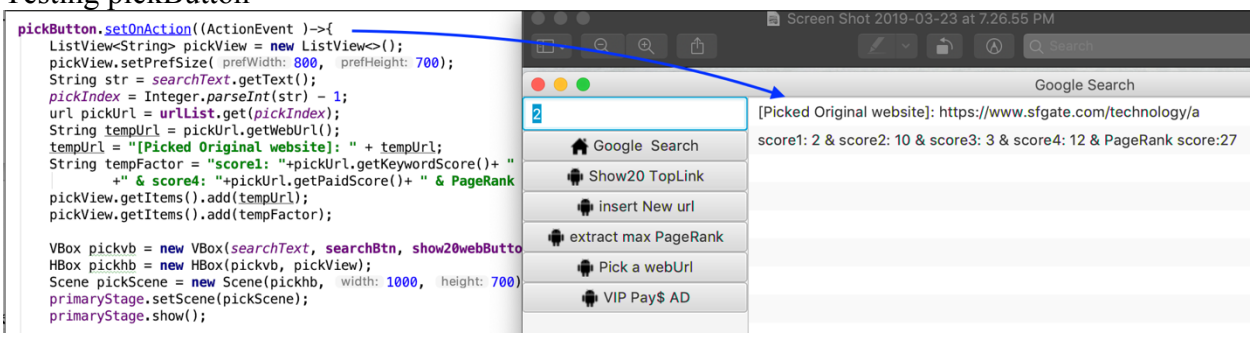
- Testing insertNewButton



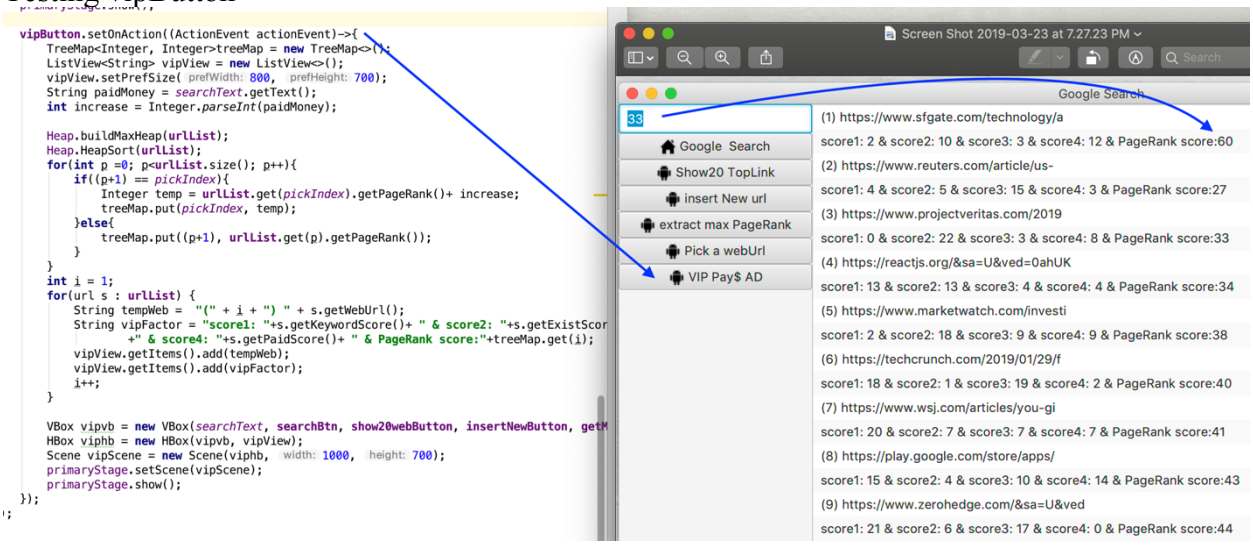
- Testing getMaxButton



- Testing pickButton

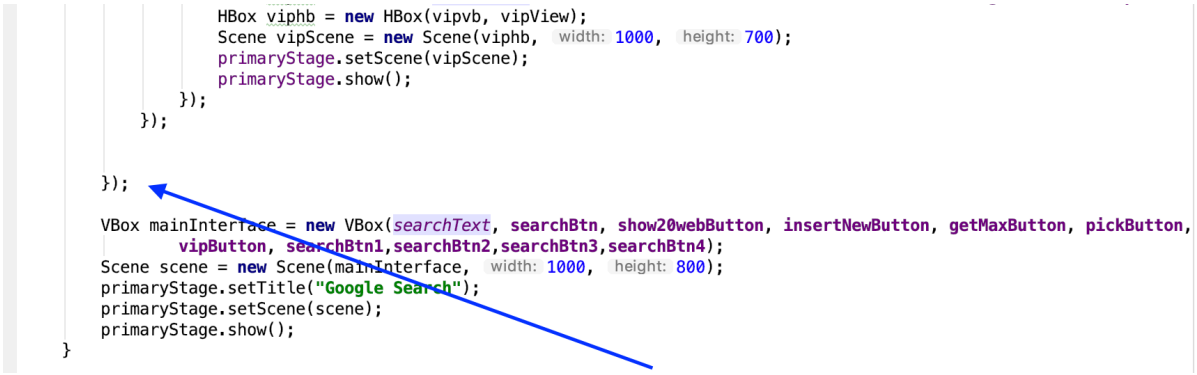


- Testing vipButton



6. Problems encountered during implementation

- Originally, I wrote all event method in Main class step by step
- → Firstly, I didn't know if the scope of searchButton will affect the following button events.
- → After struggling for more than 5 hours, I relized my mistake that the orther buttons Events must inside the searchButton's scope:



```
                HBox viphb = new HBox(vipvb, vipView);
                Scene vipScene = new Scene(viphb,  width: 1000,  height: 700);
                primaryStage.setScene(vipScene);
                primaryStage.show();
            });
        });


    });

    VBox mainInterface = new VBox(searchText, searchBtn, show20webButton, insertNewButton, getMaxButton, pickButton,
            vipButton, searchBtn1,searchBtn2,searchBtn3,searchBtn4);
    Scene scene = new Scene(mainInterface,  width: 1000,  height: 800);
    primaryStage.setTitle("Google Search");
    primaryStage.setScene(scene);
    primaryStage.show();
}
```

- **vipButton** must be inside the **pickButton**'s scope, otherwise, vipButton cannot know which index that is user picked and required.

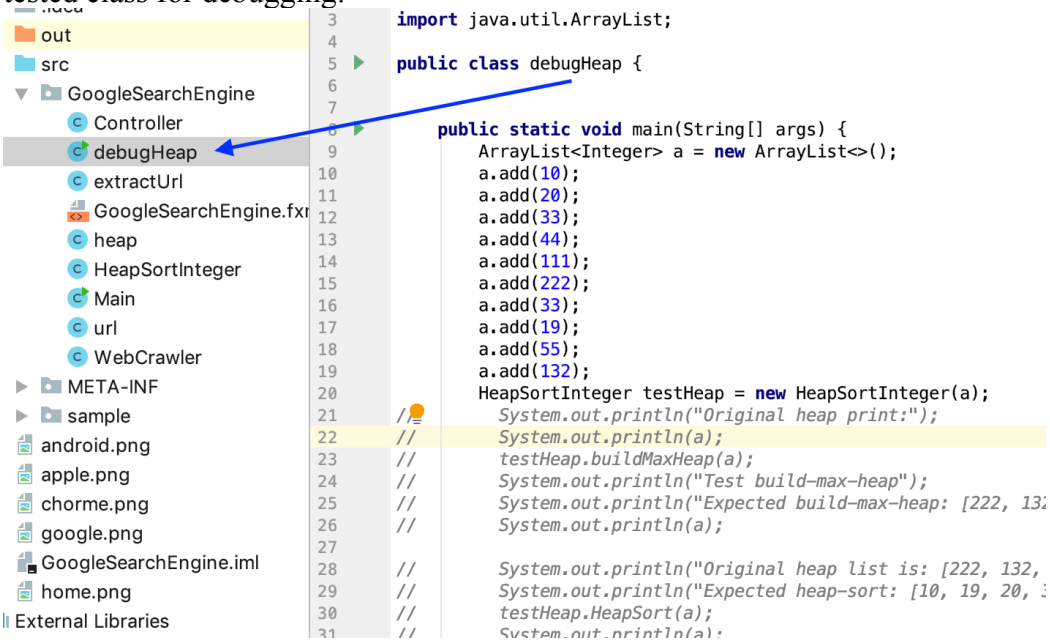
```
pickButton.setOnAction((ActionEvent )->{
    ListView<String> pickView = new ListView<>();
    pickView.setPrefSize( prefWidth: 800,  prefHeight: 700);
    String str = searchText.getText();
    pickIndex = Integer.parseInt(str) - 1;
    url pickUrl = urlList.get(pickIndex);
    String tempUrl = pickUrl.getWebUrl();
    tempUrl = "[Picked Original website]: " + tempUrl;
    String tempFactor = "score1: "+pickUrl.getKeywordScore()+ " & score2: "+pickUrl.ge
            +" & score4: "+pickUrl.getPaidScore()+ " & PageRank score:"+pickUrl.getPag
    pickView.getItems().add(tempUrl);
    pickView.getItems().add(tempFactor);

    VBox pickvb = new VBox(searchText, searchBtn, show20webButton, insertNewButton, ge
    HBox pickhb = new HBox(pickvb, pickView);
    Scene pickScene = new Scene(pickhb,  width: 1000,  height: 700);
    primaryStage.setScene(pickScene);
    primaryStage.show();

    vipButton.setOnAction((ActionEvent actionEvent)->{
        TreeMap<Integer, Integer>treeMap = new TreeMap<>();
        ListView<String> vipView = new ListView<>();
        vipView.setPrefSize( prefWidth: 800,  prefHeight: 700);
        String paidMoney = searchText.getText();
        int increase = Integer.parseInt(paidMoney);

        Heap.buildMaxHeap(urlList);
        Heap.HeapSort(urlList);
        for(int p =0; p<urlList.size(); p++){
            if((p+1) == pickIndex){
                Integer temp = urlList.get(pickIndex).getPageRank()+ increase;
```

- All buttons must be on Vbox, then the UI interface will show all buttons vertically.

```
VBox mainInterface = new VBox(searchText, searchBtn, show20webButton, insertNewButton, getMaxButton, pickButton,
        vipButton, searchBtn1,searchBtn2,searchBtn3,searchBtn4);
Scene scene = new Scene(mainInterface,  width: 1000,  height: 800);
primaryStage.setTitle("Google Search");
```

7. Leasson Learned

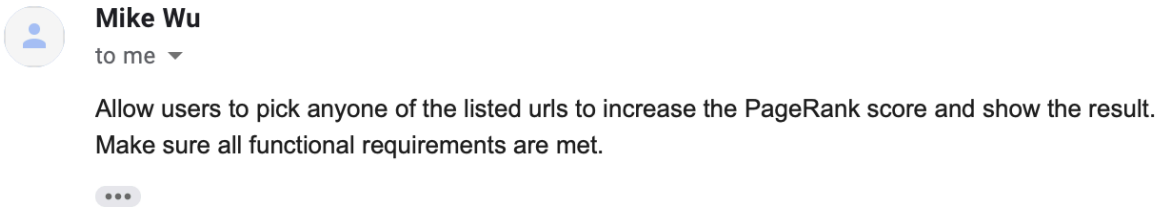
- While I encountered problems during the implementation, I learned to debug and create tested class for debugging:



- Then I find out the mistake: the arrayList must add a new key, then it will automatically Increasing it's size.

```
public void MaxHeapInsert(ArrayList<url> A, url key) {
    heapSize++;
    A.add(key);
    A.get(heapSize - 1).setPageRank((int) Double.NEGATIVE_INFINITY);
    heapIncreaseKey(A, heapSize, key);//then give the leaf real value
}
```

- Sometimes, professor Wu will help me to solve the problem that I have no idea:



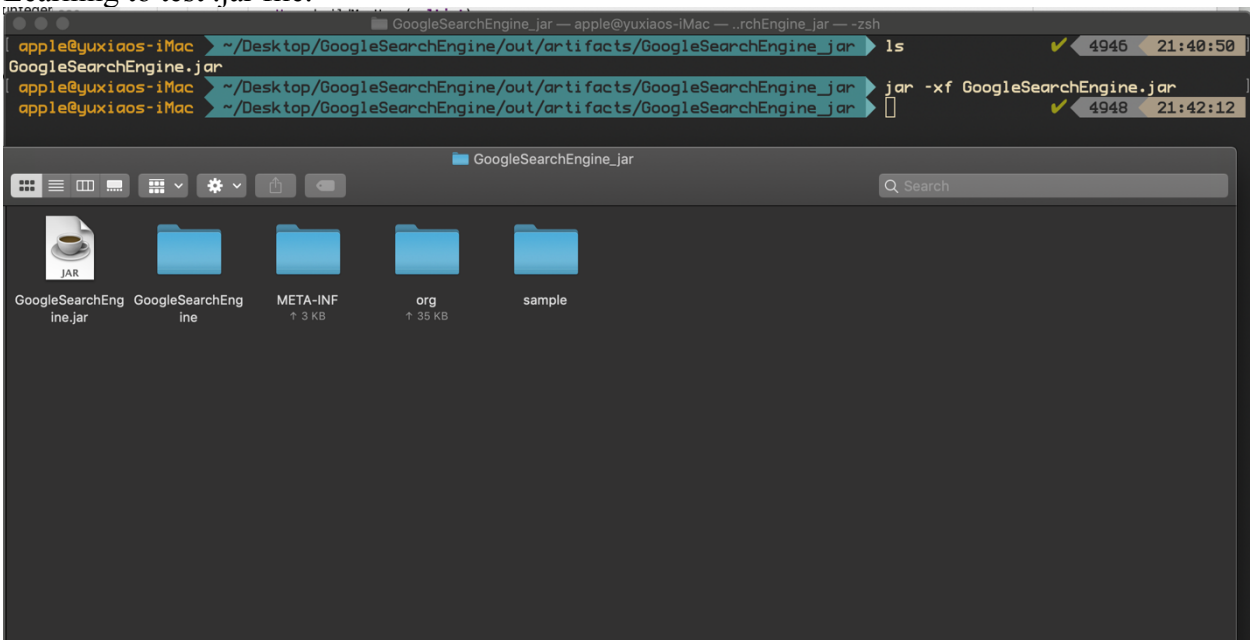
Mike Wu

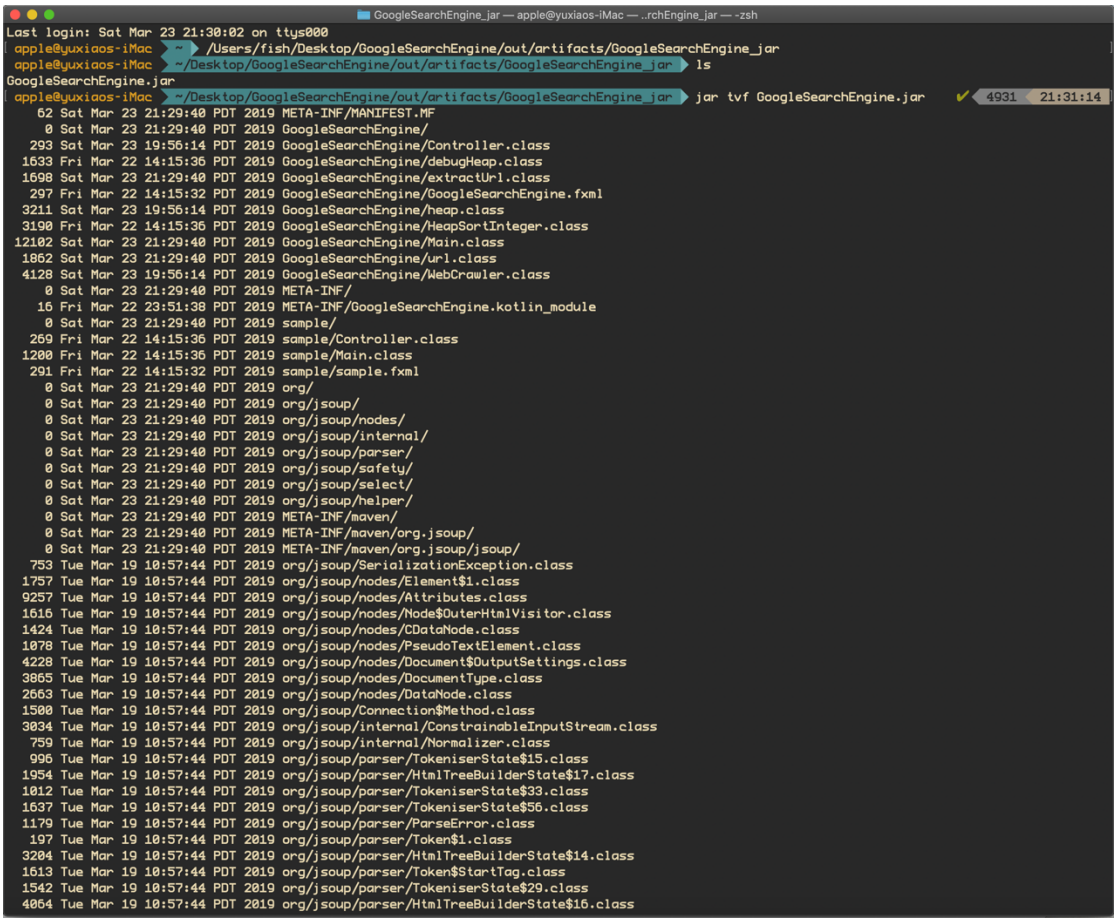
to me

Allow users to pick anyone of the listed urls to increase the PageRank score and show the result. Make sure all functional requirements are met.

- I learn to write a professional report in MS word fromat.

- Learning to test .jar file.

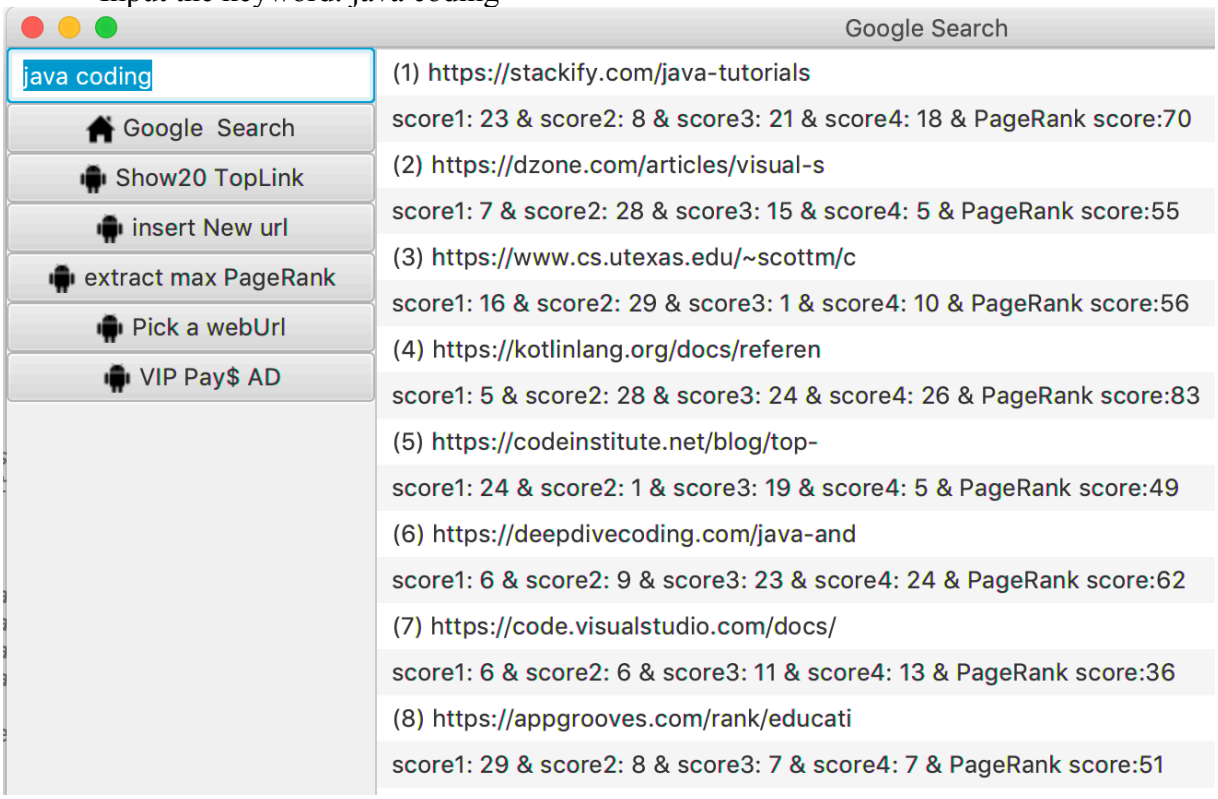




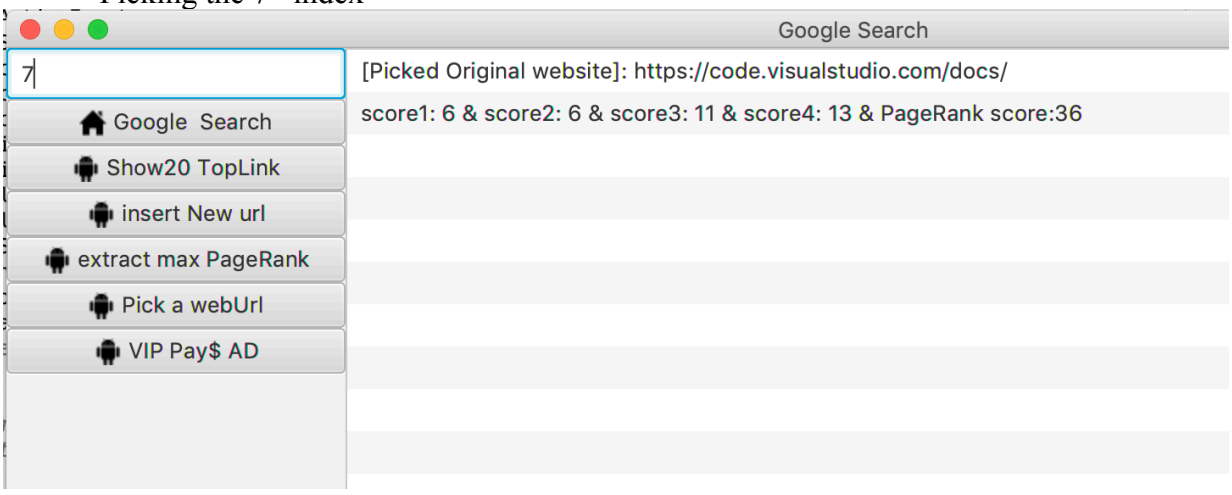
8. Updating (Debugging Report)

During last version, after increasing the score that the user picked, I forgot to resort the web url. This time I updating this new edition:

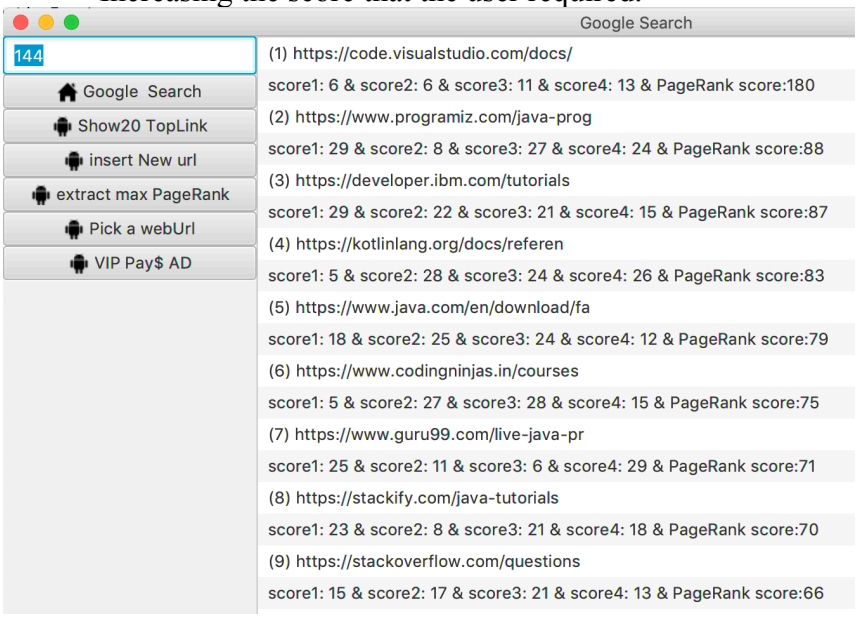
- Input the keyword: java coding



- Picking the 7th index



- Increasing the score that the user required.



- The below pictures are how to implement the requirement:

```
public url(String s){
    webUrl = s;
    linkScore = (int)(Math.random()* 30);
    keywordScore = (int)(Math.random()* 30);
    existScore = (int)(Math.random()* 30);
    score4 = (int)(Math.random()* 30);;
    paidScore = 0;

}

public url(String theUrl, int s1, int s2, int s3, int s4, int s5){
    webUrl = theUrl;
    linkScore = s1;
    keywordScore = s2;
    existScore = s3;
    score4 = s4;
    paidScore = s5;
}
```

Here, adding two constructors in order to implement the webUrl's resorted order.

```
int i = 1;
for(int v = urlList.size()-1; v>=0; v--){
    String tempWeb =  "(" + i + ") " + urlList.get(v).getWebUrl();
    String vipFactor = "score1: "+urlList.get(v).getKeywordScore()+ " & score2: "+urlList.get(v).getExistScore()+ " & score3: "+urlList.get(v).getLinkScore(
            +" & score4: "+urlList.get(v).getScore4()+ " & PageRank score:"+urlList.get(v).getPageRank();
    vipView.getItems().add(tempWeb);
    vipView.getItems().add(vipFactor);
    i++;
}

VBox vipvb = new VBox(searchText, searchBtn, show20webButton, insertNewButton, getMaxButton, pickButton,vipButton);
HBox viphb = new HBox(vipvb, vipView);
Scene vipScene = new Scene(viphb,  width: 1000,  height: 700);
primaryStage.setScene(vipScene);
primaryStage.show();
```

This for loop will help resort the web Url