BINUS UNIVERSITY INTERNATIONAL

BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter (Individual/Group* Work)

Student Information:	***Surname***	***Given Names***	***Student ID Number***
1.	Sukardi Teja	Andreas	23201900416
2.			
3.			
4.			
5.			

Course Code	: COMP6510	**Course Name**	: Programming Languages
Class	: L2AC-LEC	**Name of Lecturer(s)**	: 1. Jude Joseph Lamug Martinez 2.
Major	: Computer Science		
Title of Assignment (if any)	: Final Project		
Type of Assignment	: Project		
Submission Pattern			
Due Date	: 6/20/2020	**Submission Date**	: 6/19/2020

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating
BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality
By signing this assignment, I/we* understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I/we* declare that the work contained in this assignment is my/our* own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:	*(Name of Student)*
1.	
2. [illegible]	Andreas Sukardi Teja
3.	
4.	
5.	
etc.	

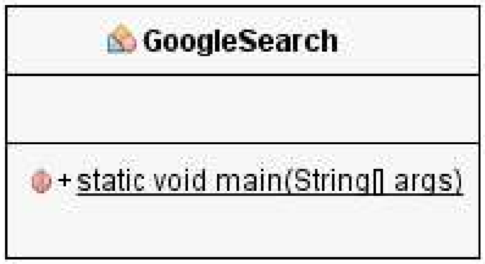
**) Delete the inappropriate option*

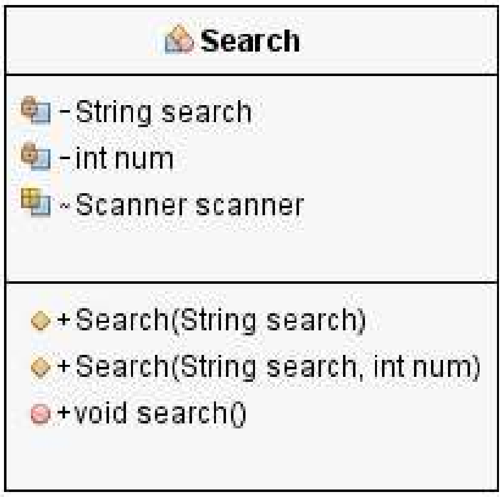
Project Specification

I didn’t exactly have a problem in mind but what I did have was a goal or a challenge to myself, which is to make a Java program capable of searching Google, extract the results, print out the results, allow the user to choose and open a link and maybe, allow the use of VPN. A search engine of sorts, or web crawler, or web scraper, not sure what the right term is for this.

Solution Design

Class Diagram

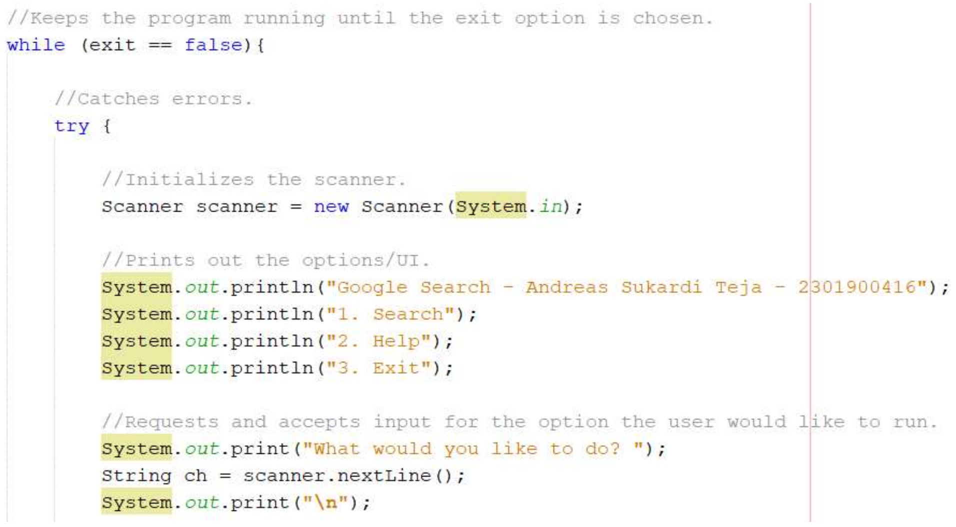




Discussion

Algorithms

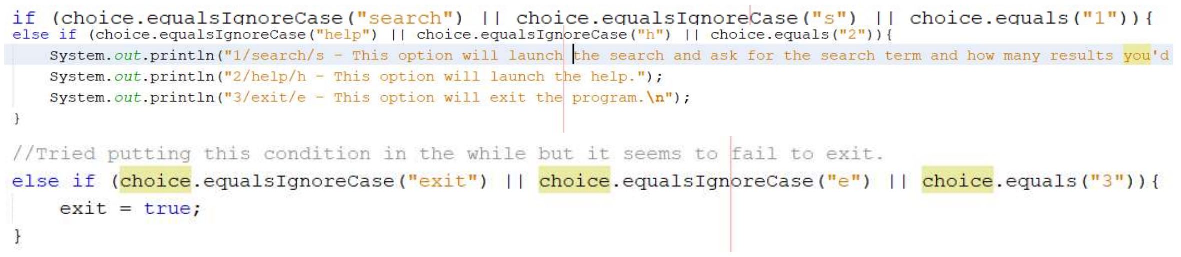
How the User Interface Works



```
//Keeps the program running until the exit option is chosen.
while (exit == false){

    //Catches errors.
    try {

        //Initializes the scanner.
        Scanner scanner = new Scanner(System.in);

        //Prints out the options/UI.
        System.out.println("Google Search - Andreas Sukardi Teja - 2301900416");
        System.out.println("1. Search");
        System.out.println("2. Help");
        System.out.println("3. Exit");

        //Requests and accepts input for the option the user would like to run.
        System.out.print("What would you like to do? ");
        String ch = scanner.nextLine();
        System.out.print("\n");
```

Figure 1.1 – The User Interface

Nothing special here, just using *while (exit == false)* to keep the program running until the *exit* flag is true. The *try-catch* is also there to catch exceptions and keep the program running. The scanner is also there to accept input from the user.

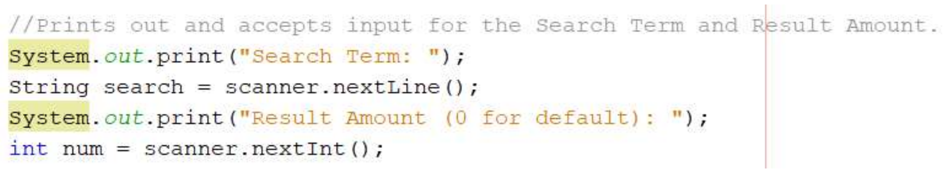


```
if (choice.equalsIgnoreCase("search") || choice.equalsIgnoreCase("s") || choice.equals("1")){
else if (choice.equalsIgnoreCase("help") || choice.equalsIgnoreCase("h") || choice.equals("2")){
    System.out.println("1/search/s - This option will launch the search and ask for the search term and how many results you'd
    System.out.println("2/help/h - This option will launch the help.");
    System.out.println("3/exit/e - This option will exit the program.\n");
}

//Tried putting this condition in the while but it seems to fail to exit.
else if (choice.equalsIgnoreCase("exit") || choice.equalsIgnoreCase("e") || choice.equals("3")){
    exit = true;
}
```

Figure 1.2 – If-Else Conditions

Simply using *If-Else* conditions here to detect the various possible valid inputs from the user with the “help” option being available if the user needs help with the input or what each option does for some reason.

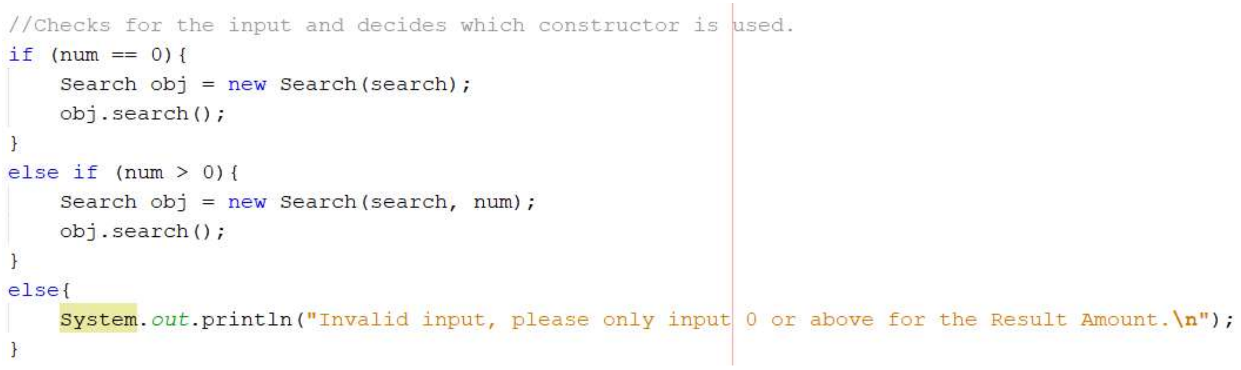
How the Input Works



```
//Prints out and accepts input for the Search Term and Result Amount.
System.out.print("Search Term: ");
String search = scanner.nextLine();
System.out.print("Result Amount (0 for default): ");
int num = scanner.nextInt();
```

Figure 2.1 – Input

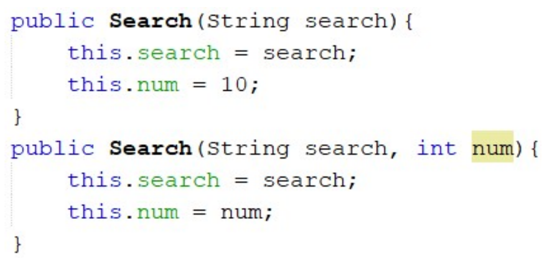
Nothing fancy here, simply using the scanner to detect the next line inputted by the user for *search* and the next integer inputted by the user for *num*.



```
//Checks for the input and decides which constructor is used.
if (num == 0){
    Search obj = new Search(search);
    obj.search();
}
else if (num > 0){
    Search obj = new Search(search, num);
    obj.search();
}
else{
    System.out.println("Invalid input, please only input 0 or above for the Result Amount.\n");
}
```

Figure 2.2 – Differentiating the Inputs

Here, I used simple *If-Else* conditions to detect which constructor the program will use, the first detecting if the input for *num* is 0 thus resulting in the program using the default value for *num* (which is 10), the second detecting if the input for *num* is greater than 0 thus resulting in the program using the inputted value as the *num* and finally, the third catching any invalid integer inputs (below 0) and giving the user an error message without causing the program to throw an exception and stop running.

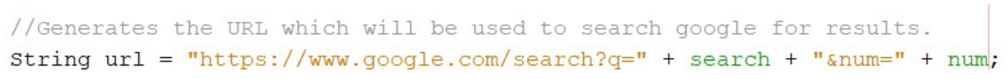


```
public Search(String search){
    this.search = search;
    this.num = 10;
}
public Search(String search, int num){
    this.search = search;
    this.num = num;
}
```

Figure 2.3 – Differing Constructors

Again, nothing fancy here, simply implementing Polymorphism to allow the option for the user to only input the search query.

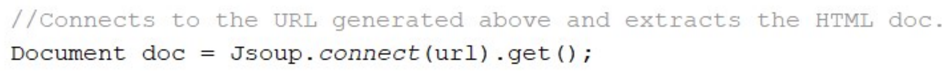
How the Search Works



```
//Generates the URL which will be used to search google for results.
String url = "https://www.google.com/search?q=" + search + "&num=" + num;
```

Figure 3.1 – Search URL

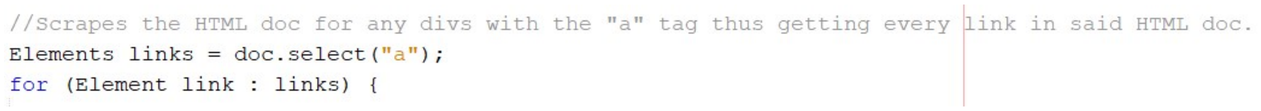
As explained by the comment lines, first the URL that will be used to do the Google search is generated by creating a string consisting of "https://www.google.com/search?q=" followed by *search*, "&num" and finally *num*. *search* in this case is the term that will be searched by Google and *num* is how many results Google will return.



```
//Connects to the URL generated above and extracts the HTML doc.
Document doc = Jsoup.connect(url).get();
```

Figure 3.2 – HTML extraction

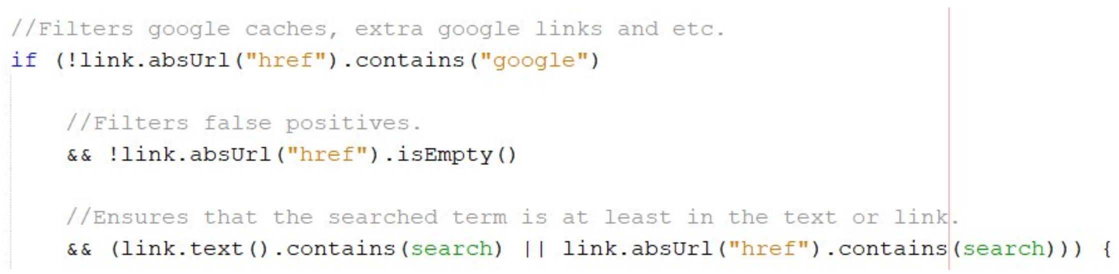
Here, *Document* simply states that *doc* is a document that will store the results of *Jsoup.connect(url).get()*. What *Jsoup.connect(url)* is doing here is connecting to the URL generated above in Figure 1.1 while *.get()* is used to extract the entirety of the HTML file.



```
//Scrapes the HTML doc for any divs with the "a" tag thus getting every link in said HTML doc.
Elements links = doc.select("a");
for (Element link : links) {
```

Figure 3.3 – Tag Filtering

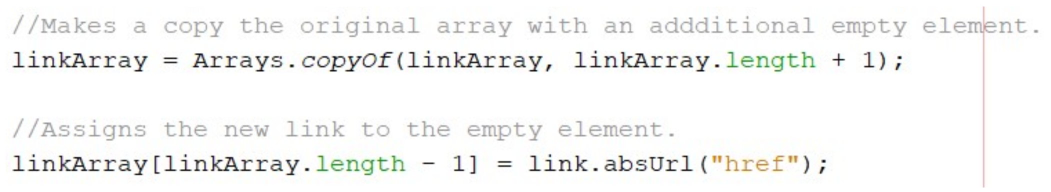
In this snippet, *.select("a")* is used to select all elements with the *a* tag and considering almost every element in the Google HTML with the *a* tag has a *href* tag right after it, it's a good bet that the elements that end up being selected have links. The *for (Element link : links)* here simply loops until all selected elements have been gone through.



```
//Filters google caches, extra google links and etc.
if (!link.absUrl("href").contains("google")

    //Filters false positives.
    && !link.absUrl("href").isEmpty()

    //Ensures that the searched term is at least in the text or link.
    && (link.text().contains(search) || link.absUrl("href").contains(search))) {
```

Figure 3.4 – If-Else Filtering

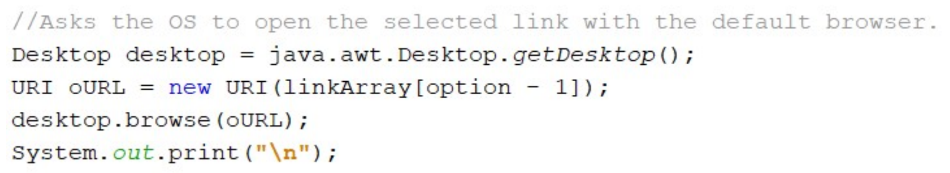
At this point I noticed that the elements I extracted included Google's other links such as their Login, Report, Images, Maps and etcetera. It included everything and this was undesirable. Some of the elements even came without links. To counter this I employed 4 *If-Else* conditions, the first ensuring that the element's absolute URL does not contain "google" thus avoiding their other unrelated links like the ones mentioned above, the second ensuring that the elements actually has links and the third and fourth ensuring that at least the element's text or absolute URL contained the search query.



```
//Makes a copy the original array with an addditional empty element.
linkArray = Arrays.copyOf(linkArray, linkArray.length + 1);

//Assigns the new link to the empty element.
linkArray[linkArray.length - 1] = link.absUrl("href");
```

Figure 3.5 – Link Storage

Since as far as I knew *.append()* didn't exist for Java arrays, I instead used *.copyOf()* which essentially makes a copy of the selected array (the first element) and allows me to change the size of the array, which for every loop would constantly make a copy of the original array with an additional empty slot so that I can store the link while there are still elements that haven't been run through by the loop in Figure 1.3.



```
//Asks the OS to open the selected link with the default browser.
Desktop desktop = java.awt.Desktop.getDesktop();
URI oURL = new URI(linkArray[option - 1]);
desktop.browse(oURL);
System.out.print("\n");
```

Figure 3.6 – Opening the Link in a Browser

This snippet of code is simply telling the OS of the computer/device to open the URL chosen by the user, this would trigger the OS to open the default browser (if it isn't already open) and open the link.

Problems

After doing a lot of research, I found out that I could use the Jsoup library to parse HTMLs but one glaring issue is that there were barely any working examples using the Jsoup library, most likely due to the examples being outdated. The only examples that worked at all were from the Jsoup documentation itself and they were quite limited. I could learn the gist of how the methods worked but I had to learn and understand Google HTML by actually reading it so I could extract the data I want. This was a major issue as the examples online simply did not work, some returning only one link while others didn't work at all, trust me, I looked. I ended up resorting to "crude" methods of *if-else* conditions to filter the links that the Google HTML would have, surprisingly the HTML had a LOT of unrelated links so it was another thing I had to learn. Despite my efforts, although the program DOES work, it's still not perfect as my current program still detects sub-links and links leading to videos though I did ensure the links are still related to the search query. Another thing I had to learn is also how Google links work as I had to know which part dictates the search query and which dictates how many results are shown.

Evidence

Code

GoogleSearch.java

```
package googlesearch;

import java.util.Scanner;

public class GoogleSearch {

        public static void main(String[] args){

            String choice = "";

            boolean exit = false;


            //Keeps the program running until the exit option is chosen.

            while (exit == false){


                //Catches errors.

                try {


                    //Initializes the scanner.

                    Scanner scanner = new Scanner(System.in);


                    //Prints out the options/UI.

                    System.out.println("Google Search - Andreas Sukardi Teja - 2301900416");

                    System.out.println("1. Search");

                    System.out.println("2. Help");

                    System.out.println("3. Exit");


                    //Requests and accepts input for the option the user would like to run.

                    System.out.print("What would you like to do? ");

                    String ch = scanner.nextLine();

                    System.out.print("\n");
            choice = ch;
            if (choice.equalsIgnoreCase("search") || choice.equalsIgnoreCase("s") || choice.equals("1")){

                //Prints out and accepts input for the Search Term and Result Amount.
                System.out.print("Search Term: ");
                String search = scanner.nextLine();
                System.out.print("Result Amount (0 for default): ");
                int num = scanner.nextInt();

                //Checks for the input and decides which constructor is used.
                if (num == 0){
                    Search obj = new Search(search);
                    obj.search();
                }
                else if (num > 0){
                    Search obj = new Search(search, num);
                    obj.search();
                }
                else{
                    System.out.println("Invalid input, please only input 0 or above for the Result
Amount.\n");
                }
            }
            else if (choice.equalsIgnoreCase("help") || choice.equalsIgnoreCase("h") ||
choice.equals("2")){
                System.out.println("1/search/s - This option will launch the search and ask for the search
term and how many results you'd like.");
                System.out.println("2/help/h - This option will launch the help.");
                System.out.println("3/exit/e - This option will exit the program.\n");
            }

            //Tried putting this condition in the while but it seems to fail to exit.
            else if (choice.equalsIgnoreCase("exit") || choice.equalsIgnoreCase("e") ||
choice.equals("3")){
                exit = true;
            }
            else {
                System.out.println("Invalid input, please type 2/help/h for the help.\n");
            }
        }
        catch(Exception e) {
            System.out.println("Invalid input, please type 2/help/h for the help.\n");
        }
    }
    System.out.println("Program Exited.");
  }
}
```

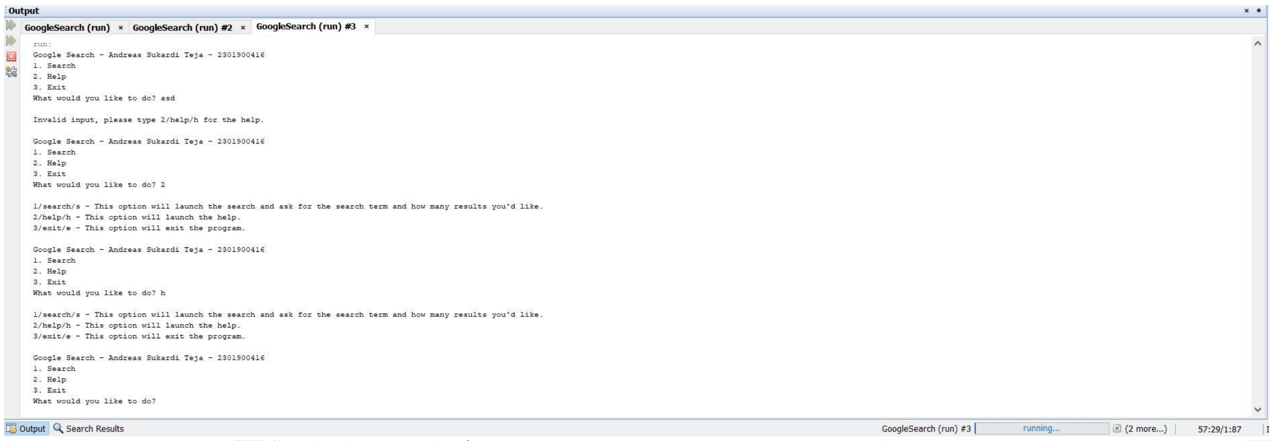
Search.java

```
package googlesearch;
import java.awt.Desktop;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.Scanner;
import java.util.Arrays;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
public class Search{
  private String search;
  private int num;
  Scanner scanner = new Scanner(System.in);
  public Search(String search){
    this.search = search;
    this.num = 10;
  }
  public Search(String search, int num){
    this.search = search;
    this.num = num;
  }
  public void search() throws IOException, URISyntaxException{

    //Catches errors.
    try {
    int count = 0;
    String[] linkArray = {};

    //Generates the URL which will be used to search google for results.
    String url = "https://www.google.com/search?q=" + search + "&num=" + num;

    //Connects to the URL generated above and extracts the HTML doc.
    Document doc = Jsoup.connect(url).get();

    //Prints out the title of the doc, in this case the search query.
    System.out.println(doc.title());
```

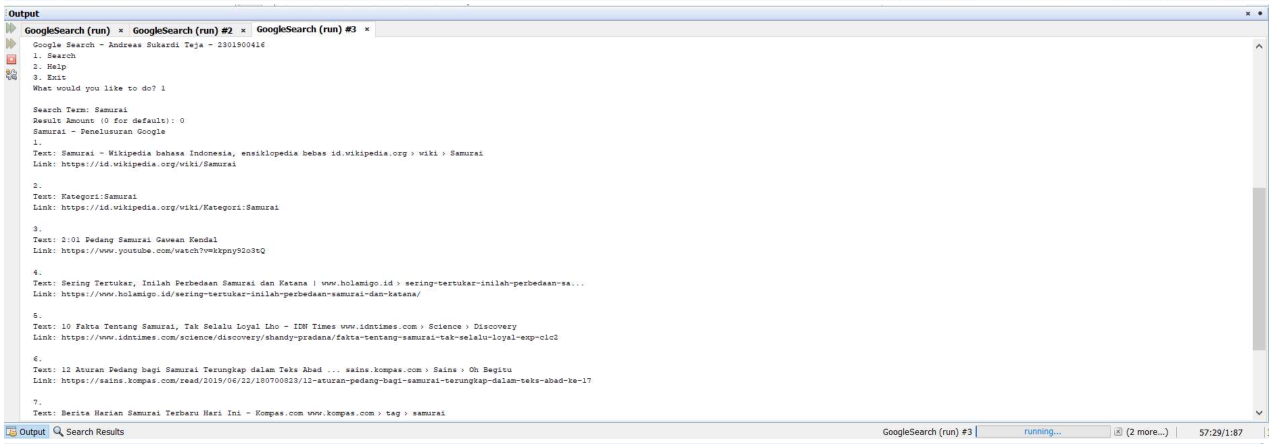
```
//Scrapes the HTML doc for any divs with the "a" tag thus getting every link in said HTML doc.

Elements links = doc.select("a");

for (Element link : links) {


   //Filters google caches, extra google links and etc.

   if (!link.absUrl("href").contains("google")


      //Filters false positives.

      && !link.absUrl("href").isEmpty()


      //Ensures that the searched term is at least in the text or link.

      && (link.text().contains(search) || link.absUrl("href").contains(search))) {


      //Keeps track and prints which number the link is.

      count = count + 1;

      System.out.println(count + ".");


      //Makes a copy the original array with an addditional empty element.

      linkArray = Arrays.copyOf(linkArray, linkArray.length + 1);


      //Assigns the new link to the empty element.

      linkArray[linkArray.length - 1] = link.absUrl("href");


      //Prints out the text and absolute URL.

      System.out.println("Text: " + link.text());

      System.out.println("Link: " + link.absUrl("href") + "\n");

   }

}
```

```
        //Prints out and accepts input for which link the user wants to open.
        System.out.print("\nWhich link would you like to open? ");
        int option = scanner.nextInt();

        //Asks the OS to open the selected link with the default browser.
        Desktop desktop = java.awt.Desktop.getDesktop();
        URI oURL = new URI(linkArray[option - 1]);
        desktop.browse(oURL);
        System.out.print("\n");

        }
        catch (Exception e){
            System.out.println("Invalid input.\n");
        }
    }
}
```

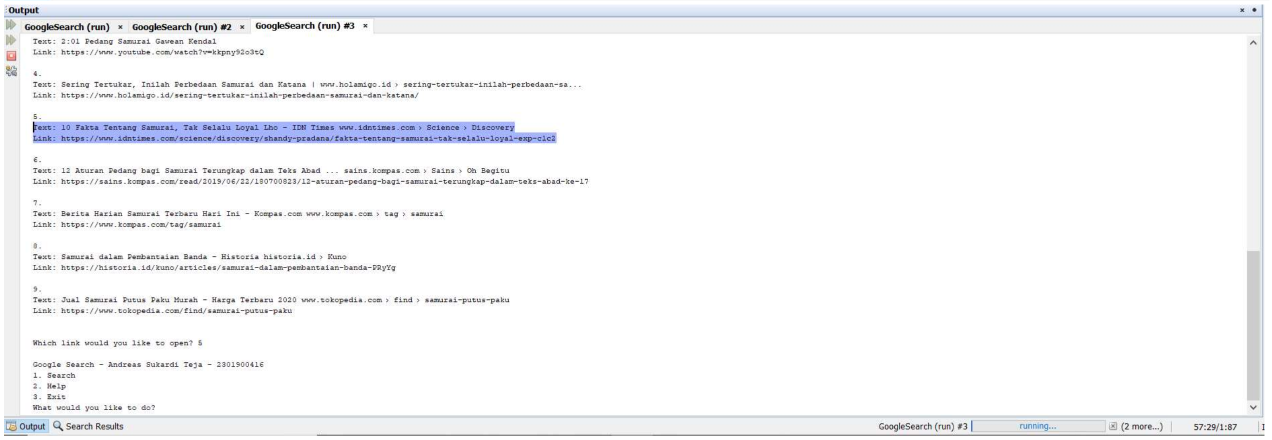
Screenshots



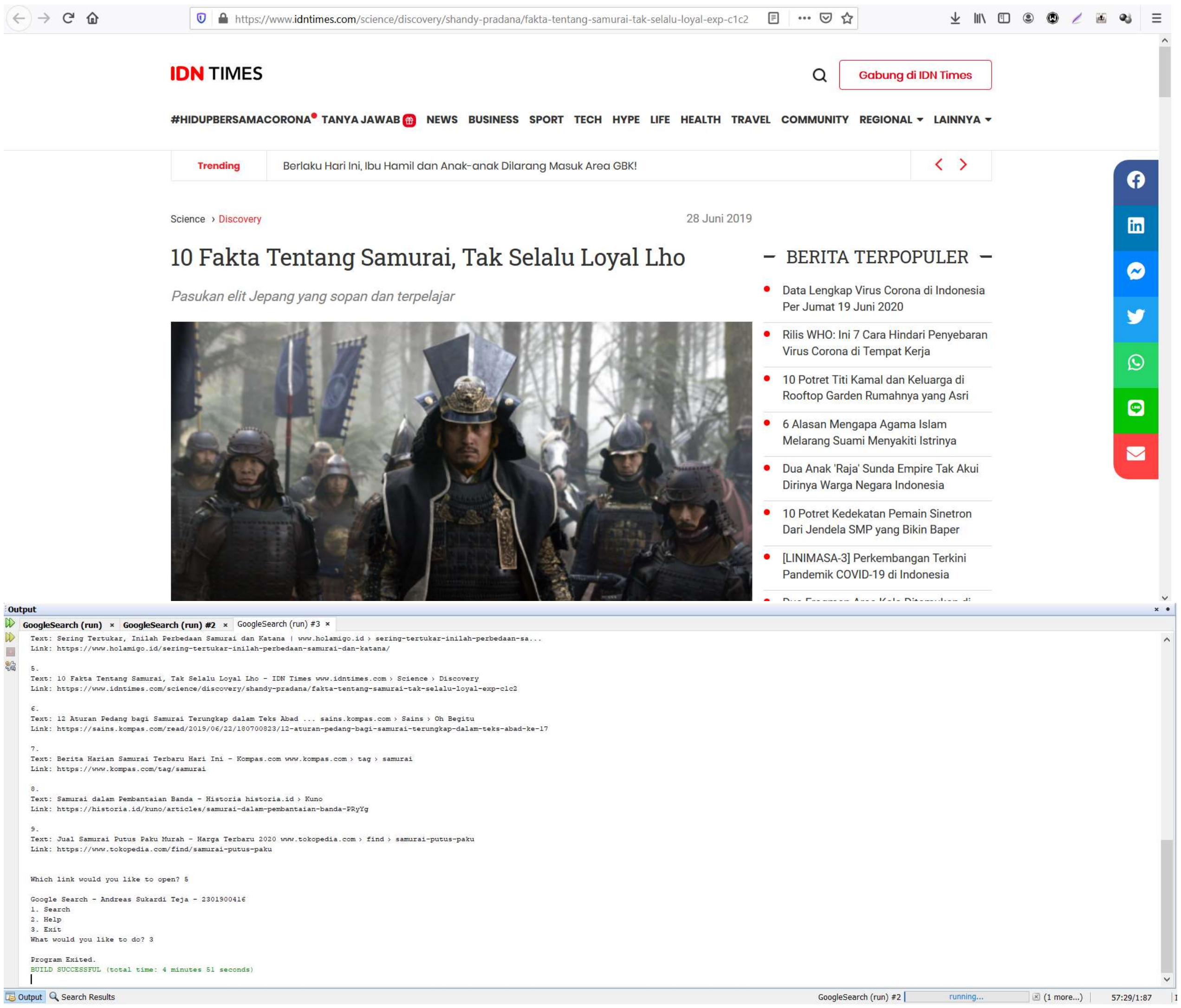
```
run:
Google Search - Andreas Sukardi Teja - 2301900416
1. Search
2. Help
3. Exit
What would you like to do? asd

Invalid input, please type 2/help/h for the help.

Google Search - Andreas Sukardi Teja - 2301900416
1. Search
2. Help
3. Exit
What would you like to do? 2

1/search/s - This option will launch the search and ask for the search term and how many results you'd like.
2/help/h - This option will launch the help.
3/exit/e - This option will exit the program.

Google Search - Andreas Sukardi Teja - 2301900416
1. Search
2. Help
3. Exit
What would you like to do? h

1/search/s - This option will launch the search and ask for the search term and how many results you'd like.
2/help/h - This option will launch the help.
3/exit/e - This option will exit the program.

Google Search - Andreas Sukardi Teja - 2301900416
1. Search
2. Help
3. Exit
What would you like to do?
```



```
Google Search - Andreas Sukardi Teja - 2301900416
1. Search
2. Help
3. Exit
What would you like to do? 1

Search Term: Samurai
Result Amount (0 for default): 0
Samurai - Penelusuran Google
1.
Text: Samurai - Wikipedia bahasa Indonesia, ensiklopedia bebas id.wikipedia.org › wiki › Samurai
Link: https://id.wikipedia.org/wiki/Samurai

2.
Text: Kategori:Samurai
Link: https://id.wikipedia.org/wiki/Kategori:Samurai

3.
Text: 2:01 Pedang Samurai Gawean Kendal
Link: https://www.youtube.com/watch?v=kkpny92o3tQ

4.
Text: Sering Tertukar, Inilah Perbedaan Samurai dan Katana | www.holamigo.id › sering-tertukar-inilah-perbedaan-sa...
Link: https://www.holamigo.id/sering-tertukar-inilah-perbedaan-samurai-dan-katana/

5.
Text: 10 Fakta Tentang Samurai, Tak Selalu Loyal Lho - IDN Times www.idntimes.com › Science › Discovery
Link: https://www.idntimes.com/science/discovery/shandy-pradana/fakta-tentang-samurai-tak-selalu-loyal-exp-c1c2

6.
Text: 12 Aturan Pedang bagi Samurai Terungkap dalam Teks Abad ... sains.kompas.com › Sains › Oh Begitu
Link: https://sains.kompas.com/read/2019/06/22/180700823/12-aturan-pedang-bagi-samurai-terungkap-dalam-teks-abad-ke-17

7.
Text: Berita Harian Samurai Terbaru Hari Ini - Kompas.com www.kompas.com › tag › samurai
```



```
Text: 2:01 Pedang Samurai Gawean Kendal
Link: https://www.youtube.com/watch?v=kkpny92o3tQ

4.
Text: Sering Tertukar, Inilah Perbedaan Samurai dan Katana | www.holamigo.id › sering-tertukar-inilah-perbedaan-sa...
Link: https://www.holamigo.id/sering-tertukar-inilah-perbedaan-samurai-dan-katana/

5.
Text: 10 Fakta Tentang Samurai, Tak Selalu Loyal Lho - IDN Times www.idntimes.com › Science › Discovery
Link: https://www.idntimes.com/science/discovery/shandy-pradana/fakta-tentang-samurai-tak-selalu-loyal-exp-c1c2

6.
Text: 12 Aturan Pedang bagi Samurai Terungkap dalam Teks Abad ... sains.kompas.com › Sains › Oh Begitu
Link: https://sains.kompas.com/read/2019/06/22/180700823/12-aturan-pedang-bagi-samurai-terungkap-dalam-teks-abad-ke-17

7.
Text: Berita Harian Samurai Terbaru Hari Ini - Kompas.com www.kompas.com › tag › samurai
Link: https://www.kompas.com/tag/samurai

8.
Text: Samurai dalam Pembantaian Banda - Historia historia.id › Kuno
Link: https://historia.id/kuno/articles/samurai-dalam-pembantaian-banda-PRyYg

9.
Text: Jual Samurai Putus Paku Murah - Harga Terbaru 2020 www.tokopedia.com › find › samurai-putus-paku
Link: https://www.tokopedia.com/find/samurai-putus-paku


Which link would you like to open? 5

Google Search - Andreas Sukardi Teja - 2301900416
1. Search
2. Help
3. Exit
What would you like to do?
```



https://www.idntimes.com/science/discovery/shandy-pradana/fakta-tentang-samurai-tak-selalu-loyal-exp-c1c2

IDN TIMES

Gabung di IDN Times

#HIDUPBERSAMACORONA TANYA JAWAB NEWS BUSINESS SPORT TECH HYPE LIFE HEALTH TRAVEL COMMUNITY REGIONAL LAINNYA

Trending Berlaku Hari Ini, Ibu Hamil dan Anak-anak Dilarang Masuk Area GBK!

Science › Discovery 28 Juni 2019

10 Fakta Tentang Samurai, Tak Selalu Loyal Lho

Pasukan elit Jepang yang sopan dan terpelajar

BERITA TERPOPULER

- Data Lengkap Virus Corona di Indonesia Per Jumat 19 Juni 2020
- Rilis WHO: Ini 7 Cara Hindari Penyebaran Virus Corona di Tempat Kerja
- 10 Potret Titi Kamal dan Keluarga di Rooftop Garden Rumahnya yang Asri
- 6 Alasan Mengapa Agama Islam Melarang Suami Menyakiti Istrinya
- Dua Anak 'Raja' Sunda Empire Tak Akui Dirinya Warga Negara Indonesia
- 10 Potret Kedekatan Pemain Sinetron Dari Jendela SMP yang Bikin Baper
- [LINIMASA-3] Perkembangan Terkini Pandemik COVID-19 di Indonesia

Output

GoogleSearch (run) GoogleSearch (run) #2 GoogleSearch (run) #3

```
Text: Sering Tertukar, Inilah Perbedaan Samurai dan Katana | www.holamigo.id › sering-tertukar-inilah-perbedaan-sa...
Link: https://www.holamigo.id/sering-tertukar-inilah-perbedaan-samurai-dan-katana/

5.
Text: 10 Fakta Tentang Samurai, Tak Selalu Loyal Lho - IDN Times www.idntimes.com › Science › Discovery
Link: https://www.idntimes.com/science/discovery/shandy-pradana/fakta-tentang-samurai-tak-selalu-loyal-exp-c1c2

6.
Text: 12 Aturan Pedang bagi Samurai Terungkap dalam Teks Abad ... sains.kompas.com › Sains › Oh Begitu
Link: https://sains.kompas.com/read/2019/06/22/180700823/12-aturan-pedang-bagi-samurai-terungkap-dalam-teks-abad-ke-17

7.
Text: Berita Harian Samurai Terbaru Hari Ini - Kompas.com www.kompas.com › tag › samurai
Link: https://www.kompas.com/tag/samurai

8.
Text: Samurai dalam Pembantaian Banda - Historia historia.id › Kuno
Link: https://historia.id/kuno/articles/samurai-dalam-pembantaian-banda-PRyYg

9.
Text: Jual Samurai Putus Paku Murah - Harga Terbaru 2020 www.tokopedia.com › find › samurai-putus-paku
Link: https://www.tokopedia.com/find/samurai-putus-paku


Which link would you like to open? 5

Google Search - Andreas Sukardi Teja - 2301900416
1. Search
2. Help
3. Exit
What would you like to do? 3

Program Exited.
BUILD SUCCESSFUL (total time: 4 minutes 51 seconds)
```

Output Search Results GoogleSearch (run) #2 running... (1 more...) 57:29/1:87

Resources

https://jsoup.org/

http://zetcode.com/java/jsoup/

https://www.tutorialspoint.com/jsoup/index.htm

https://www.journaldev.com/7144/jsoup-java-html-parser

https://www.codeproject.com/Questions/398241/how-to-open-url-in-java