Project Report

Even Semester (2024)

Binus university

Binus international

Object Oriented Programming final project (Individual work)

Student Information:

Surname: Marlent **Given Name:** Steven gerald **Student ID:**2702398283

Course Code : COMP6699001 **Course Name :** Object Oriented Programming

Class : L2AC **Lecturer :** Jude Joseph Lamug Martinez, MCS

Type of Assignment : Final Project Report

Submission Pattern

Due Date : 20 June 2024

Submission Date : 20 June 2024

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 offences 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 understand, accept, and consent to Binus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my 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:

Steven Gerald Marlent

TABLE OF CONTENT

Project Specification

Overview

This program is a Multithreaded web crawler. The web crawler will fetch web pages starting from a given URL and follow links within those pages up to a specified depth. Each crawler will run in its own thread to enable concurrent fetching of web pages.

1. Program Input:
   List of URLs for web crawling and it's hard coded in the 'Main' class
2. Program output:
   Messages indicating the creation of web crawler instances.
   Notifications when a webpage is successfully fetched, including the bot ID and the URL.
   The title of the fetched webpage and the list of URLs the crawler has visited

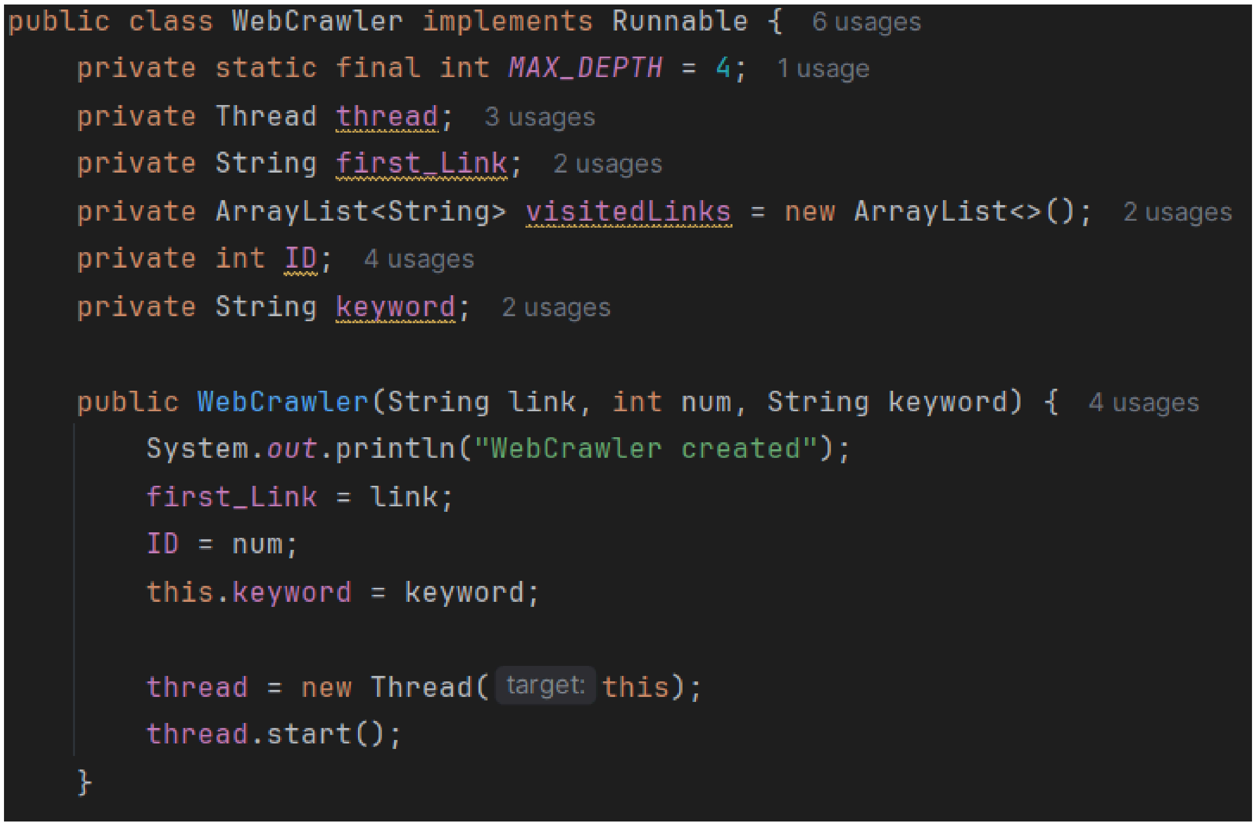
Library used

1. ArrayList
   - Used store URLs to be visited
2. Exception
   - used to handle input/output errors that may occur when reading from or writing to files
3. Jsoup
   Used to connects to an URL and fetches its content
   - jsoup.Connection: allows to use the "Connection" class from the Jsoup library
   - jsoup.nodes.Document: Used to navigate and manipulate the document's elements and attributes.
   - jsoup.nodes.Element: Used to access and manipulate the element's attributes
   - jsoup.select.Elements: Used to selecting multiple elements from a document using CSS selectors

Solution Design

Files involved in this program are Main.java and WebCrawler.java

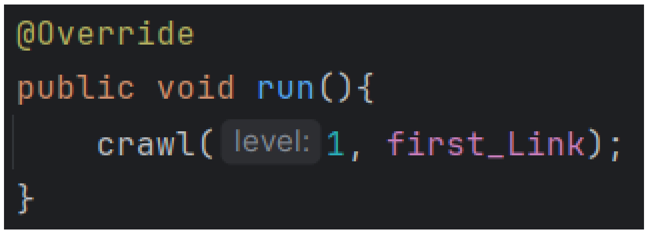
WebCrawler.java



```java
public class WebCrawler implements Runnable {  6 usages
    private static final int MAX_DEPTH = 4;  1 usage
    private Thread thread;  3 usages
    private String first_Link;  2 usages
    private ArrayList<String> visitedLinks = new ArrayList<>();  2 usages
    private int ID;  4 usages
    private String keyword;  2 usages

    public WebCrawler(String link, int num, String keyword) {  4 usages
        System.out.println("WebCrawler created");
        first_Link = link;
        ID = num;
        this.keyword = keyword;

        thread = new Thread( target: this);
        thread.start();
    }
```

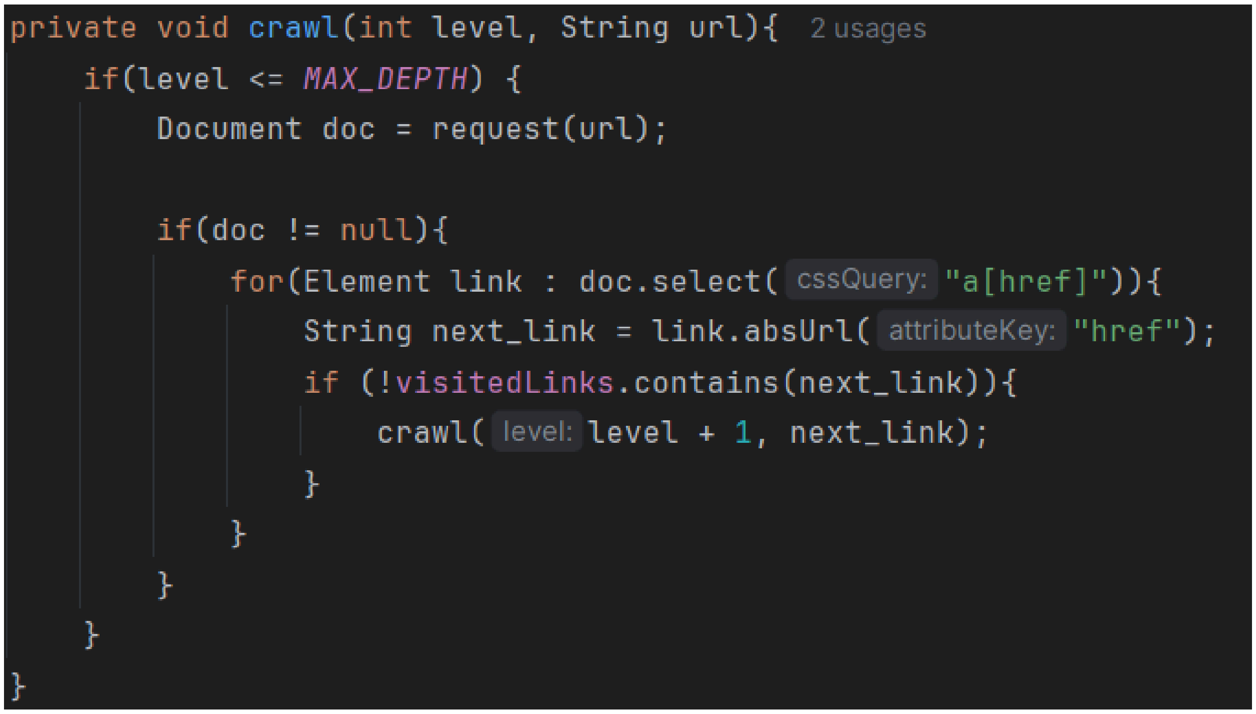
This class is implemented by ‘runnable’ which must provide an implementation for the ‘run’ method so the web crawler can be executed by a thread. Then declaring the static constant field (MAX_DEPTH) that defines the maximum depth to which the web crawler will recurse, field (thread) holds the Thread instance that will run this WebCrawler instance, field stores the initial URL that the web crawler will start from, field (visitedLinks) that keeps track of all the URLs that the web crawler has visited to avoid revisiting them, field (ID) stores a unique identifier for each WebCrawler instance, and field (keyword) stores a keyword used to filter URLs.

The ‘WebCrawler’ constructor called 3 parameters that start the URL for the web crawler, the unique identifier for the web crawler instance, and the keyword used to filter URLs. Inside the constructor has printed that the webcrawler has been started, Initializes the ‘first_Link’ field with the provided starting URL, Initializes the ‘ID’ field with the provided identifier, Initializes the ‘keyword’ field with the provided keyword, Creates a new ‘Thread’ object, passing this as the target. This means the ‘run’ method of this instance will be executed when the thread is started, Starts the ‘thread’, which triggers the execution of the run method in a new thread of execution.



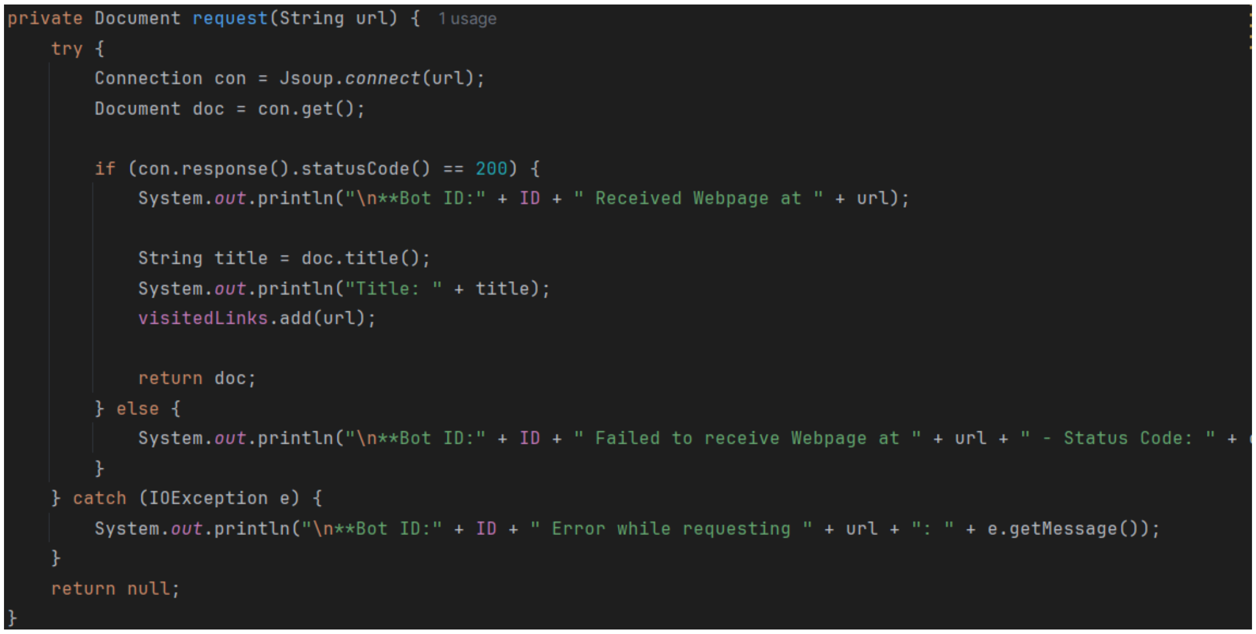
```
@Override
public void run(){
    crawl( level: 1, first_Link);
}
```

This run method initiates the web crawling process by calling the crawl method. 1 means the first level of crawling and first_link is the string variable that holds the URL from which the crawling process will begin.



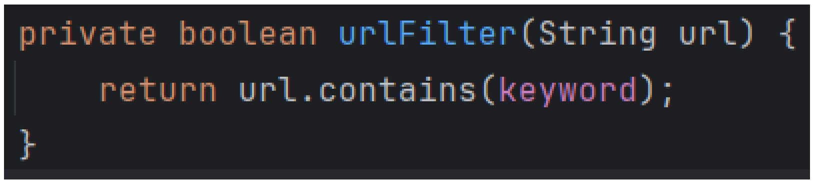
```
private void crawl(int level, String url){  2 usages
    if(level <= MAX_DEPTH) {
        Document doc = request(url);

        if(doc != null){
            for(Element link : doc.select( cssQuery: "a[href]")){
                String next_link = link.absUrl( attributeKey: "href");
                if (!visitedLinks.contains(next_link)){
                    crawl( level: level + 1, next_link);
                }
            }
        }
    }
}
```

In this crawl method first it checks If the depth exceeds ‘MAX_DEPTH’, the method returns immediately, preventing further recursion and thereby limiting how deeply the crawler will navigate through links. Then it calls a method request with the current URL. The request method presumably makes an HTTP request to the URL and retrieves the HTML document, which is then stored in doc. If the document successfully retrieved it will extract links then will proceed to convert the relative URL in the href attribute to an absolute URL. afterwards the condition check if next_link has already been visited (to avoid reprocessing the same link) and whether it passes a URL filter which might be used to ensure the link meets certain criteria If the link hasn't been visited and passes the filter, the crawl method is called recursively with an incremented depth and the new URL



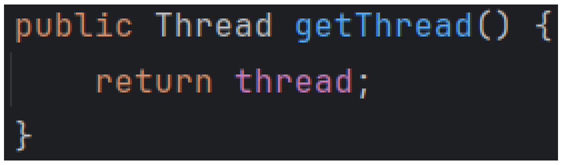
```java
private Document request(String url) {  1 usage
    try {
        Connection con = Jsoup.connect(url);
        Document doc = con.get();

        if (con.response().statusCode() == 200) {
            System.out.println("\n**Bot ID:" + ID + " Received Webpage at " + url);

            String title = doc.title();
            System.out.println("Title: " + title);
            visitedLinks.add(url);

            return doc;
        } else {
            System.out.println("\n**Bot ID:" + ID + " Failed to receive Webpage at " + url + " - Status Code: " +
        }
    } catch (IOException e) {
        System.out.println("\n**Bot ID:" + ID + " Error while requesting " + url + ": " + e.getMessage());
    }
    return null;
}
```

This request method establishes a connection to a specified URL. Fetch the web page's HTML content. Verify the HTTP response status code. If the page is successfully retrieved, it prints a success message with the bot ID and URL aftwards prints the title of the web page. Adds the URL to the list of visited links. Returns the Document object containing the page's HTML content. If the retrieval fails or an exception occurs, it prints an error message. Then returns null.



```java
private boolean urlFilter(String url) {
    return url.contains(keyword);
}
```

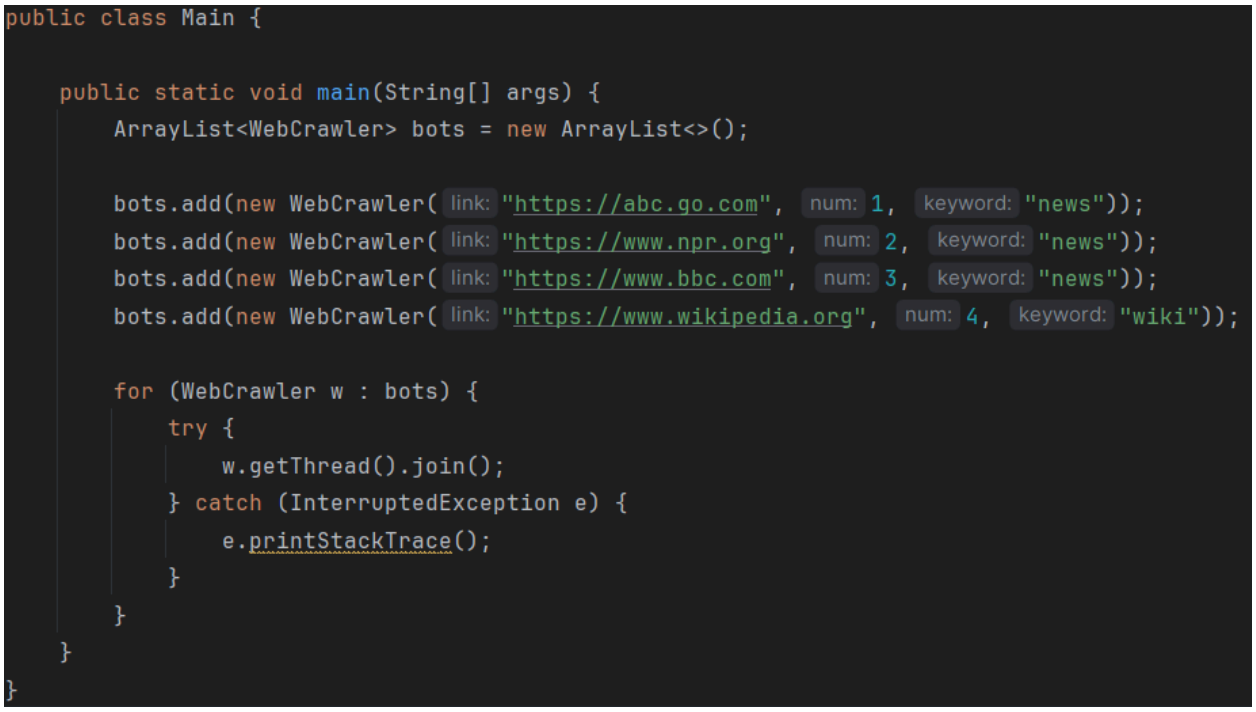
This urlFilter method checks whether a given URL contains a specific keyword. This is used to filter and decide which URLs the web crawler should process further. If the URL contains the keyword, the method returns true; otherwise, it returns false



```java
public Thread getThread() {
    return thread;
}
```

The ‘getThread’ method in the ‘WebCrawler’ class provides access to the ‘Thread’ object associated with a ‘WebCrawler’ instance. This allows external code to interact with the thread, such as waiting for it to complete or checking its status.

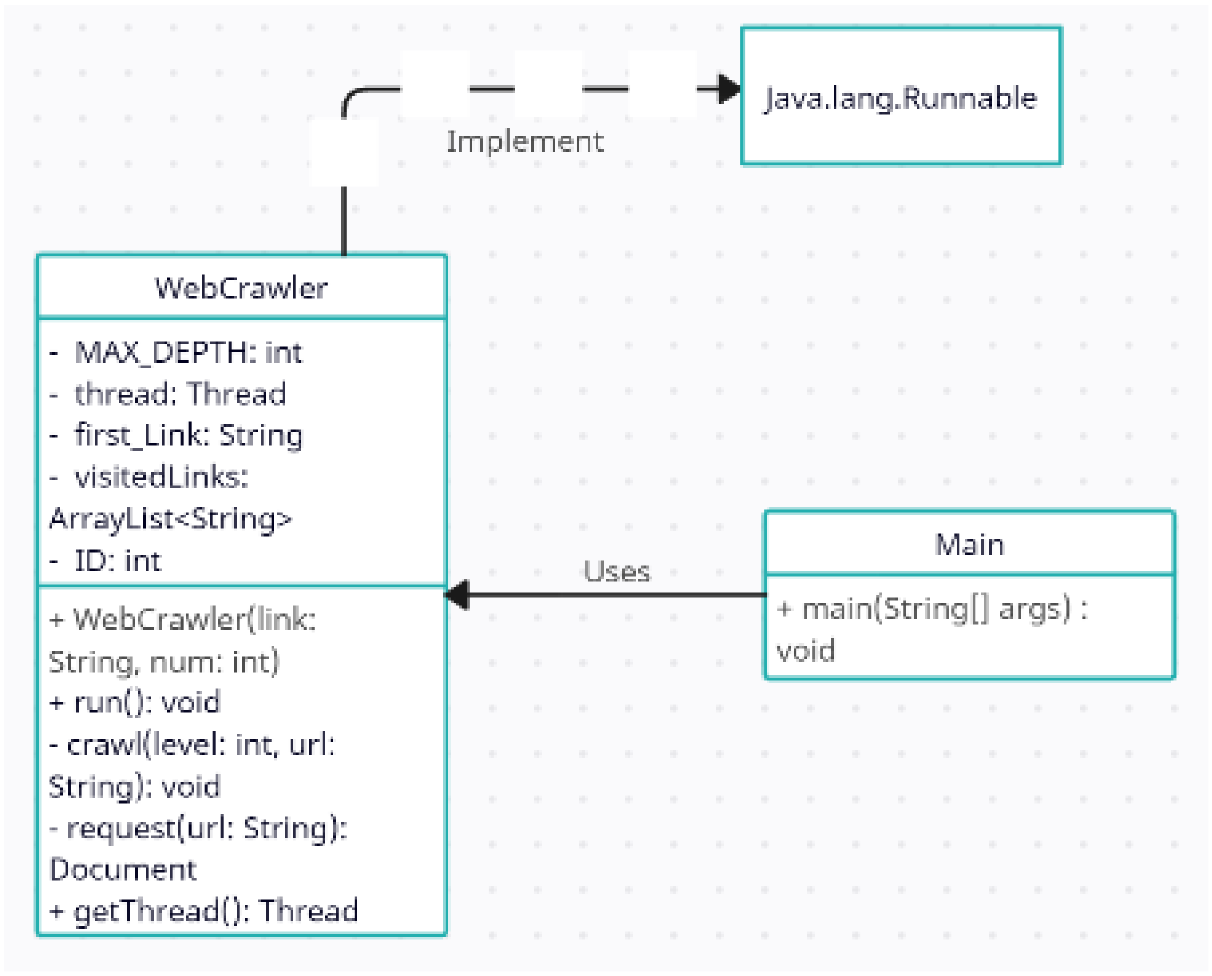
Main.java



```java
public class Main {

    public static void main(String[] args) {
        ArrayList<WebCrawler> bots = new ArrayList<>();

        bots.add(new WebCrawler( link: "https://abc.go.com",  num: 1,  keyword: "news"));
        bots.add(new WebCrawler( link: "https://www.npr.org",  num: 2,  keyword: "news"));
        bots.add(new WebCrawler( link: "https://www.bbc.com",  num: 3,  keyword: "news"));
        bots.add(new WebCrawler( link: "https://www.wikipedia.org",  num: 4,  keyword: "wiki"));

        for (WebCrawler w : bots) {
            try {
                w.getThread().join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

The main class Creates an ArrayList to hold WebCrawler instances. Adds four WebCrawler instances to the list, each initialized with a different URL, ID, and keyword. Waits for all WebCrawler threads to complete using the join method, ensuring that the main method does not exit until all crawling operations are finished.

Class Diagram


Java.lang.Runnable
Implement
WebCrawler
- MAX_DEPTH: int
- thread: Thread
- first_Link: String
- visitedLinks: ArrayList<String>
- ID: int
+ WebCrawler(link: String, num: int)
+ run(): void
- crawl(level: int, url: String): void
- request(url: String): Document
+ getThread(): Thread
Uses
Main
+ main(String[] args) : void

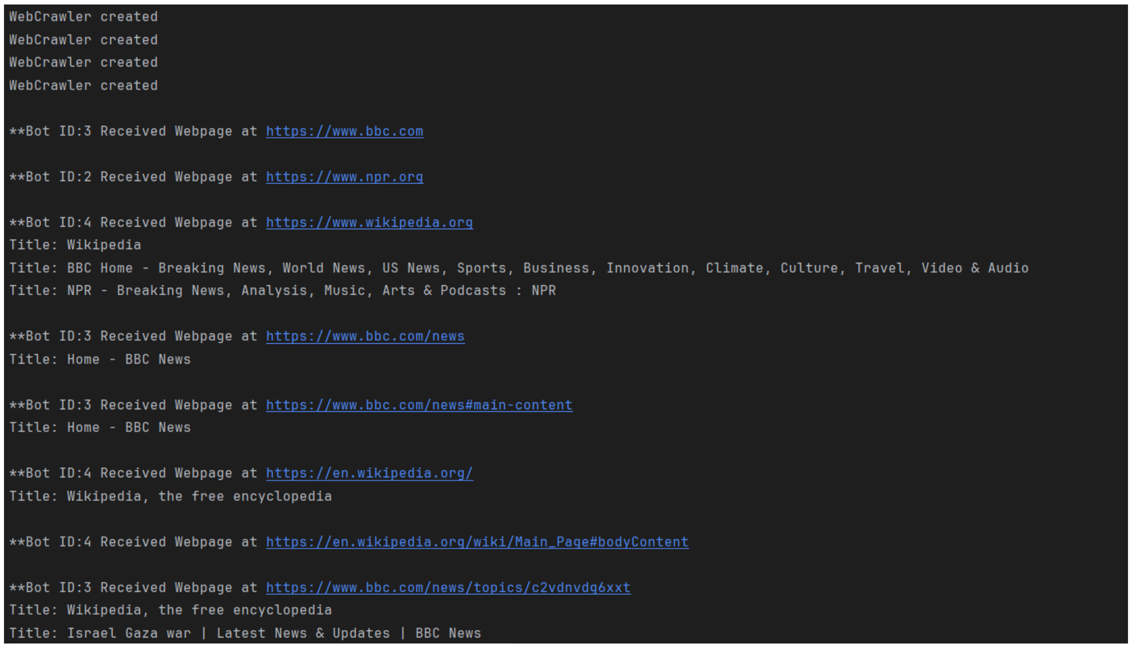
How it works

1. Initialize Main Class:
    - The Main class is defined with the main method as the entry point of the application.
2. Create a List for WebCrawlers:
    - An ArrayList named bots is created to hold multiple WebCrawler instances.
3. Instantiate WebCrawlers:
    - Four WebCrawler objects are created with different URLs, IDs, and keywords.
    - Each WebCrawler instance starts its own thread upon creation.
4. Manage Threads:
    - A loop iterates over the WebCrawler instances in the bots list.
    - For each WebCrawler, the main thread waits for its associated thread to finish using the join() method. This ensures that the main method does not exit until all crawling operations are completed.
5. WebCrawler Class Setup
    - The WebCrawler class implements the Runnable interface.
    - It has member variables for maximum depth, thread, initial URL, visited links, ID, and keyword.
6. WebCrawler Constructor:
    - Initializes the WebCrawler instance with the given URL, ID, and keyword.
    - Creates and starts a new thread for the web crawler.
7. Run Method:
    - The run method is executed when the thread starts.
    - It calls the crawl method with the initial URL and a starting depth of 1.
8. Crawl Method:
    - The crawl method performs the web crawling operation.
    - It fetches the content of the given URL.
    - If the content is successfully fetched, it processes all the links found on the page.
    - For each link, if it has not been visited and it contains the keyword, the method recursively calls itself with the next depth level and the new URL.
    - The recursion continues until the maximum depth (MAX_DEPTH) is reached.
9. Request Method:

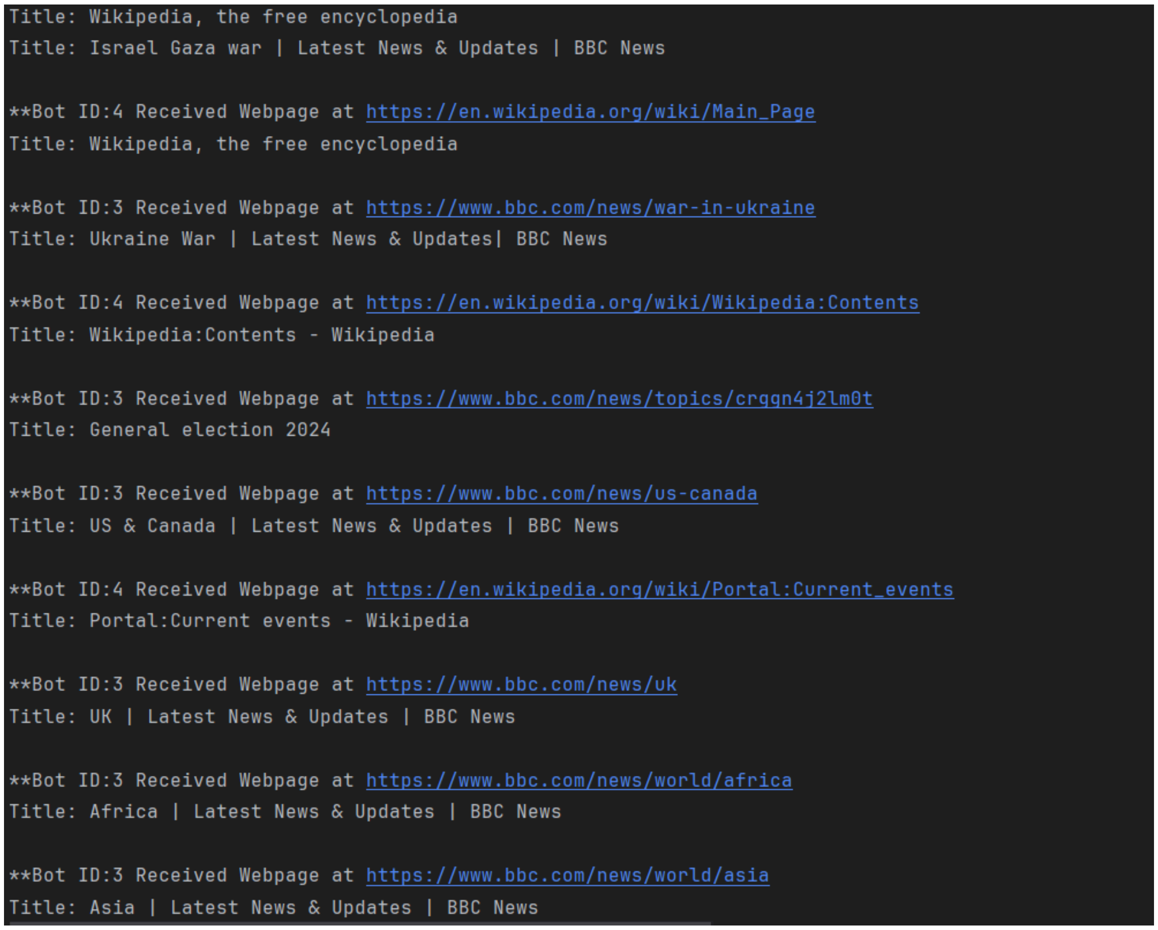
- The request method uses Jsoup to connect to the URL and retrieve the web page content.
- If the request is successful (HTTP status code 200), it prints the page title and adds the URL to the list of visited links.
- If the request fails, it prints an error message.

10. URL Filter Method:
    - The urlFilter method checks if the given URL contains the specified keyword.
    - Returns true if the keyword is found, otherwise returns false.
11. Get Thread Method:
    - The getThread method returns the thread associated with the WebCrawler.

Evidence of Working Program



```
WebCrawler created
WebCrawler created
WebCrawler created
WebCrawler created

**Bot ID:3 Received Webpage at https://www.bbc.com

**Bot ID:2 Received Webpage at https://www.npr.org

**Bot ID:4 Received Webpage at https://www.wikipedia.org
Title: Wikipedia
Title: BBC Home - Breaking News, World News, US News, Sports, Business, Innovation, Climate, Culture, Travel, Video & Audio
Title: NPR - Breaking News, Analysis, Music, Arts & Podcasts : NPR

**Bot ID:3 Received Webpage at https://www.bbc.com/news
Title: Home - BBC News

**Bot ID:3 Received Webpage at https://www.bbc.com/news#main-content
Title: Home - BBC News

**Bot ID:4 Received Webpage at https://en.wikipedia.org/
Title: Wikipedia, the free encyclopedia

**Bot ID:4 Received Webpage at https://en.wikipedia.org/wiki/Main_Page#bodyContent

**Bot ID:3 Received Webpage at https://www.bbc.com/news/topics/c2vdnvdg6xxt
Title: Wikipedia, the free encyclopedia
Title: Israel Gaza war | Latest News & Updates | BBC News
```



```
Title: Wikipedia, the free encyclopedia
Title: Israel Gaza war | Latest News & Updates | BBC News

**Bot ID:4 Received Webpage at https://en.wikipedia.org/wiki/Main_Page
Title: Wikipedia, the free encyclopedia

**Bot ID:3 Received Webpage at https://www.bbc.com/news/war-in-ukraine
Title: Ukraine War | Latest News & Updates| BBC News

**Bot ID:4 Received Webpage at https://en.wikipedia.org/wiki/Wikipedia:Contents
Title: Wikipedia:Contents - Wikipedia

**Bot ID:3 Received Webpage at https://www.bbc.com/news/topics/crggn4j2lm0t
Title: General election 2024

**Bot ID:3 Received Webpage at https://www.bbc.com/news/us-canada
Title: US & Canada | Latest News & Updates | BBC News

**Bot ID:4 Received Webpage at https://en.wikipedia.org/wiki/Portal:Current_events
Title: Portal:Current events - Wikipedia

**Bot ID:3 Received Webpage at https://www.bbc.com/news/uk
Title: UK | Latest News & Updates | BBC News

**Bot ID:3 Received Webpage at https://www.bbc.com/news/world/africa
Title: Africa | Latest News & Updates | BBC News

**Bot ID:3 Received Webpage at https://www.bbc.com/news/world/asia
Title: Asia | Latest News & Updates | BBC News
```

Video demo

https://drive.google.com/file/d/1OHsD-6hQx3Eqy35hGJE5pfCJ9VlVa-4n/view?usp=sharing

Resources

What is a web crawler? | How web spiders work | Cloudflare. (n.d.). https://www.cloudflare.com/learning/bots/what-is-a-web-crawler/

W3Schools.com. (n.d.). https://www.w3schools.com/java/java_arraylist.asp

S, R. A. (2023, February 21). *Learn multithreading in Java with examples*. Simplilearn.com. https://www.simplilearn.com/tutorials/java-tutorial/multithreading-in-java

https://youtu.be/r_MbozD32eo?si=KDN4asSBaIKyZIz9