CS 172
Introduction to Information Retrieval
Project Phase 1

Team Member:

- Akiyo Yokota

- Vincent Pang

1. Collaboration Details

We divided our work 50/50, with details below:

Akiyo:

- Designed the main crawling method in pseudo code
- Designed all the code relative to reading/writing/file/directory functions in utility class.
- Designed all the crawling history/check duplication related code
- Designed handling robots.txt related code

Vincent:

- Designed the method of extracting URL from a web page
- Designed all the objects in the project, they include:
    - META
    - NormalizedUrl
    - Pair
    - Robot
- Designed all the crawling webpage related code. This includes checking if a webpage is HTML; does it returns status of 200; how to fetch the page into memory etc
- Putting the main method together with the method we already own.

2. Overview of System

- Architecture & Crawling Strategy

  Pseudo code for main crawling method:

```
Queue <= seeds.LoadURL()

While( !Queue.empty() && numPageCrawled < numPageAllowed) {

        url = Queue.pop()

        if(!url.isCrawlable()) //details explained below

                skip

        url.downloadPage()

        url.downloadRobot.txt()

        links <= url.extractLink()
```

```
        Queue <= links.filter() //details explained below
    }
```

Pseudo code for check if a url is crawlable:

```
    if(depth > numHops)  return false;
    if(url.connectionStatus() != 200) return false;
    if(!url.isHTML()) return false;
    if(url.isDup()) return false;
    return true;
```

Pseudo code for filtering links:

```
    links.removeDup() //remove any links that's been crawled already
    if(robots.txt == null)   queue <= links
    else      queue <= links.followRobotsRules()
```

- Data Structure
  - Pair : <url : String , depth : Integer>
  - META : <noindex : bool, index : bool, follow : bool, nofollow : bool>
  - NormalizedUrl : <protocol : String, port : int, host : String, path : String, query : String, bookmark : String, url : String>
  - Robot : <UserAgent : String, metaContent : map<String, META>, crawl_delay : int>
  - history : List<String>
  - urlQueue : Queue<Pair>

3. Limitation of System

- Can do:
  - Identify duplicate url
  - Able to remember what's been crawled when restart the program
  - Able avoid program being hanged with time out
  - Able to identify broken link
  - Able to download the content of link and write them into files
  - Able to identify if a page has robots and follow it's rules
  - Able to start the program with a set of seeds from a file

    - Able to link url to the location of downloaded html file
    - Able to set a limit on number of pages to crawl each run
    - Able to set a limit on depth of web page to crawl.
- Can't do:
    - Can't Identify duplicated content
    - Can't run in threads

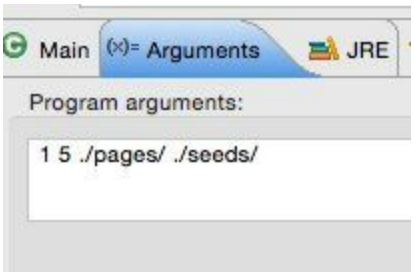
4. How to Deploy System

- To Build:
    - The project uses Maven to handle dependencies, use maven install to download the dependency.
    - The project was developed using Eclipse, recommend using Eclipse to run the project.
- To Run:
    - Place crawler.sh and CS172_Crawler_ayoko001_vpang002.jar in the same directory
    - Create a directory for seed files
        1. Put all seed files in this directory
    - Create a directory where the outputs are to be saved
    - There are four parameters:
        1. Number of hops from the base
        2. Number of pages to crawl for current execution
        3. Location of directory to download html file (must contain '/' at the end)
        4. Location of directory that contains seeds (must contain '/' at the end)
    - Usage: sh ./crawler.sh <hops-away: 6> <num-pages: 10000> <seed-dir:./seeds/> <output-dir:./outputs/>

5. Screenshots showing system in action

Scenario 1: limit number of page to 5:

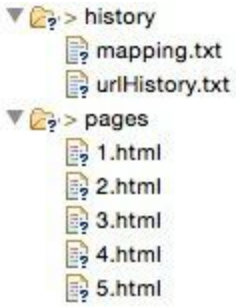
With this parameter



This will be the output:

```
URL: http://www.cs.ucr.edu       Depth: 0
URL: http://www1.cs.ucr.edu/index.php   Depth: 0
URL: http://www1.cs.ucr.edu/     Depth: 0
URL: http://www.ucr.edu/         Depth: 0
URL: http://www.engr.ucr.edu/    Depth: 0
```

These files will be generated:



mapping.txt shows you where is the html file corresponding to the url

```
1 http://www.cs.ucr.edu:./pages/1.html
2 http://www1.cs.ucr.edu/index.php:./pages/2.html
3 http://www1.cs.ucr.edu/:./pages/3.html
4 http://www.ucr.edu/:./pages/4.html
5 http://www.engr.ucr.edu/:./pages/5.html
6
```

And urlHistory.txt will record the pages we've crawl

```
1 http://www.cs.ucr.edu
2 http://www1.cs.ucr.edu/index.php
3 http://www1.cs.ucr.edu/
4 http://www.ucr.edu/
5 http://www.engr.ucr.edu/
6
```

Scenario 2: limit number of page to 5000 action in terminal: