Exercise 2

Java OOP, Threads, a bit of Http

- **נביאים**:
  - https://classroom.github.com/a/XpMVWSvD

  תאריך הגשה: 23:59 6/5/2024
- **שטראוס נשים**:
  - https://classroom.github.com/a/Sp1yPUt3

  תאריך הגשה: 23:59 9/5/2024
- **שטראוס גברים**:
  - https://classroom.github.com/a/2WUaE4NQ

  תאריך הגשה: 23:59 6/5/2024

In this exercise, you will build a simple image crawler that takes as input URLs, and produces an output with results.

Requirements:

1. Input: a TEXT file containing one URL per line. If the file is not accessible it displays an error message and exits. There should be NO assumptions on the file name format (abc, abc.txt, abc.def are valid file names).
2. The program receives 3 mandatory arguments (remember that the program name is NOT part of the args in Java):
    1. The output format: a combination of characters among {`s,u,t,` m} where
        i. `s` is size: the size of image given by content-length header of the http response
        ii. `u` is url: the URL itself
        iii. `t` is time: the time (in milliseconds) for the resource to respond to a http get request
        iv. `m` is the image type: the mime type as given in the content-type header of the http response
    2. a pool size (positive non zero number, see below),
    3. an input file name. The file path may be relative or absolute.
3. If any argument is missing or invalid it displays a usage message and exits
4. The program spins off one thread per URL to be processed.
5. Use a thread pool: limit the the number of simultaneous threads according to pool size argument
6. The URL should be analyzed and only URL of images will be processed. There should not be any duplicates URL in the output. The output should be printed in the same order as the input file.

Program Arguments

The main class of your program should be named *Main*.
For example executing the program with "*us 5 1000 urls.txt*" as program arguments (in IntelliJ you can enter them in your run configuration) will execute the program as follow (java is the Java interpreter, Main is your program main class):

java Main ut 5 urls.txt

means the program will output url and time, handle a pool size of 5 threads, and the file containing the urls is named urls.txt.

Output

The program output the results, according to the required format (program parameter 1), for example if the parameter specifies “sutm” (size,url,time,mime type) it prints:

1002 http://a.b.c/logo.jpeg 500 image/jpeg
4598 http://a.b.c/abc 4300 image/gif

The output format maybe any combination of the 4 letters or less, in any order, with at least one letter (“usm”, “muts”, “u”, “sm” are valid combinations)

Note the possible cases:

- URL already processed, skip it
- Success opening the URL, print the required output
- Malformed URL: displaying “malformed” to System.err for example: x.com/gh malformed
- Unreachable URL: “failed” to System.err for example: http://a.b.c/def failed

Design and implementation remarks:

HTTP

In order to find out if the URL is an image, you will look at the *Content-Type* header of the HTTP response (it should begin with “`image/`”). See the HttpUrlConnection Java class (getContentType() method).

Similarly, in order to get the size of the resource, you will check the *Content-Length* header and if not present, will return -1.

The thread pool

You may either implement the thread pool yourself, or (recommended) use the ready made Java classes – (see an explanation and example here).

Critical sections

Race conditions (if any) must be handled precisely. You will be penalized for useless/overdoing critical sections, or missing critical sections. Note that some Collections classes are already synchronized but it will protect atomic actions on the data structure (add, remove etc…). In other words, using these classes does not mean you are solving the race conditions.

Good OOP

Your code is expected to be built with OOP concepts such as the single responsibility principle. Use any relevant design patterns among the one you are expected to know (such as singleton, factory, visitor, decorator, strategy, observer) or others if you wish. A good design means that your code is open for extension but closed for modification. Note also that simple inheritance is not the answer to all.

Address at least 3 of the following requirements :

1. Ability to deal with additional type of output information
2. Optimizing memory usage
3. Reading input from another source (for example from terminal instead of file)
4. Processing other types of resource (not just images)

You may suggest other requirements instead of the one listed above, in such case consult with the teacher.
OOP design will weight 20% of the grade.

Submission

1. Read the syllabus to review general requirements
2. Describe in the README which design patterns you have used, and which problem it solves.
3. Add full comments even for private methods (generate the Javadoc API – make sure to enable private methods Javadoc in your project - and link the generated index.html in your README.html).
4. Submit at least one test input file, named urls.txt.

Good luck!