Assignment 3: Thread Pooling Implementation

Students: Joanna C. S. Santos and Yue Hua

1. Domain

The application is a Web crawler that reports the status of the links contained in a Web page. It detects all links referenced by a Web page, i. e., those that are contained within the `href` attribute of anchor tags (`<a>`) and prints a report of the status (404 - Not found, 200 - OK, 500 - Internal Server Error, etc) for each of them. Thus, as input, the application receives a URL to a Web page - as a JAVA argument - and outputs the HTTP status code from performing a HEAD request to each of the found links in the Web page, as shown in Figure 1.

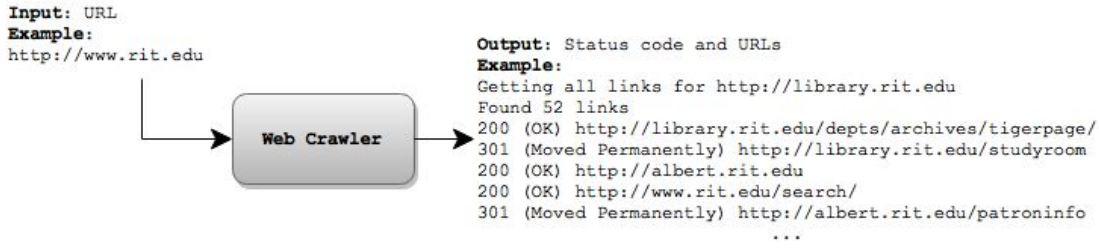

Figure 1: Web crawler input and outputs example

2. Performance critical task

The overall algorithm of the application is:

1. Perform HTTP GET request to the specified URL
2. Search for anchor tags (*<a>*)
3. For each anchor tag found:
   a. Extract the link contained in the *href* attribute
   b. Perform a HTTP HEAD request to the link found
   c. Print response status code followed by the URL of the link

The task performed in 3.b is time consuming, since it requires performing a HTTP request to the Web server, which potentially can last for a few seconds. Hence, in order to produce faster results, we created a pool of threads that performs the steps 3.b and 3.c.

3. Design

As shown in Figure 2, to implement the Web crawler with thread pooling, we designed the `WebCrawler` class that has the `execute()` method, that triggers the crawling of the Web page, and has a reference to an `ExecutorService` which is the thread pool.

This pool of threads manages the execution of the objects of the class. The `Task` class is a concrete implementation of the `Runnable` interface. The `run()` method of this class execute the task of performing a HTTP HEAD request to a link and prints the respective status code of the response.

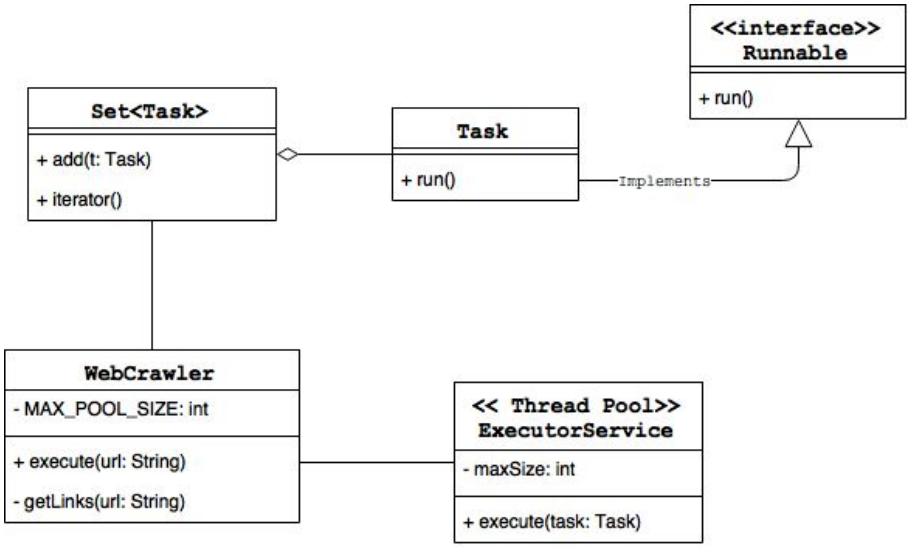

Figure 2: Class diagram for the Web crawler