HW 6 Web Crawler

이번 과제는 제11강의 내용을 기초로 하여 Web Crawler를 작성하는 것입니다. 이 프로그램은 특정 웹 페이지에서 출발하여 그 페이지가 참조하는 다른 페이지들의 웹 주소(URL)들을 수집하기 위한 것입니다.
이 프로그램은 웹 페이지 접속을 위한 HttpURLConnection, 입출력을 위한 Stream과 Reader, 문자열 처리를 위한 String의 의 여러 기능, 열어 본 페이지와 수집된 주소들을 관리하기 위한 컬렉션(List와 Set 등), 예외처리 등이 복합적으로 사용되는 프로그램입니다.

프로그램의 기본적인 구성은 다음과 같습니다.

1. 커맨드 라인에 시작 페이지의 웹 주소(URL)와 탐색을 원하는 최대 깊이(depth)를 위한 정수를 입력하여 다른 웹 페이지들에 대한 참조의 탐색을 시작합니다. 필요한 인자들이 제대로 제공된 경우 다음 명령을 사용하여 프로그램을 실행할 수 있으며, 인자의 개수가 틀린 경우 올바른 사용법을 출력한 후 즉시 실행을 종료해야 합니다.

   java Crawler <URL> <depth>

   커맨드 라인 인자 처리 방법은 https://www.tutorialspoint.com/Java-command-line-arguments을 참고하기 바랍니다.
2. 주어진 URL String과 depth 0을 처리해야할 페이지 정보의 초기값으로 저장하여야 합니다. URL과 depth의 쌍은 프로그램 전반에 걸쳐 사용되므로 이를 **UrlDepthPair**라는 클래스를 제공하니 이를 활용하기 바랍니다.
3. 처리해야할 페이지들의 리스트에 남은 항목이 있는 동안 다음 작업을 계속합니다.
   A. 하나의 항목 [aURL, aDepth]을 리스트에서 제거합니다.
   B. aURL을 위한 HttpURLConnection 연결을 얻고, 이로부터 InputStream과 Reader를 구합니다.
   C. Reader로부터 한 라인을 읽고 다른 웹 페이지에 대한 참조를 포함하고 있는지 확인합니다. 다른 웹 페이지에 대한 참조는 <a href="**http...**">와 같은 문자열로 구성됩니다. 여기서 "**http...**" 부분을 수집하면 됩니다. 한 라인에 여러 개의 참조가 있을 수도 있습니다. 한 참조가 여러 라인에 걸쳐 표현된 경우는 처리하지 않아도 됩니다. 발견된 URL newURL은 aDepth가 최대 깊이보다 작은 값이고 이전에 처리한 적이 없으면 처리해야할 페이지들의 리스트와 결과 리스트에 [newURL, aDepth+1]로 저장합니다.
4. 단계 3이 종료하면 결과 리스트에 포함된 모든 항목들([URL, depth])을 출력합니다.

웹 사이트 접속과 데이터 전송(입력)에 소요되는 시간에 따라 프로그램의 동작이 크게 좌우될 수 있습니다. 특정 사이트에 접속하거나 데이터를 읽어올 때 너무 오래 걸리면

예외를 발생시키도록 하여 그 페이지를 포기하는 것이 자연스럽습니다. 이를 위해 HttpURLConnection conn을 구한 후 실제 입력을 요청하기 전 적절한 위치에 다음 코드를 추가하여 접속과 입력을 위해 각각 최대 1초와 3초까지만 대기하도록 할 수 있습니다.

```
conn.setConnectTimeout(1000);
conn.setReadTimeout(3000);
```

다수의 웹 페이지들은 HTTP가 아니라 HTTPS 기반입니다. HTTPS 기반의 웹 서버는 서버 인증서를 클라이언트에 전달하여 그 인증서가 올바른 인증서인지 검사하도록 요청합니다. 이 기능을 올바르게 작성하려면 인터넷 보안에 대한 깊은 지식이 필요합니다. 본 과제에서는 다음과 같은 코드를 main() 메서드에서 웹 페이지 탐색을 시작하기 전 적당한 위치에 추가하여 서버에서 보내오는 모든 인증서를 신뢰하도록 할 수 있습니다.

```
// Set up SSL context for HTTPS support
try {
    TrustManager[] trustAllCerts =
        new TrustManager[]{ new X509TrustManager() {
            public X509Certificate[] getAcceptedIssuers() { return null; }
            public void checkClientTrusted(X509Certificate[] certs, String authType) {}
            public void checkServerTrusted(X509Certificate[] certs, String authType) {}
        }};
    SSLContext sc = SSLContext.getInstance("SSL");
    sc.init(null, trustAllCerts, new SecureRandom());
    HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
} catch (Exception e) { System.exit(1); }
```

지금까지 학습한 내용들을 종합적으로 활용해야 하는 과제여서 다소 어려워 보일 수 있지만, 과제를 완료하고 나면 이 정도의 노력만으로도 이런 수준의 문제를 해결할 수 있다는 것에 Java에 대한 매력을 크게 느낄 수 있을 것입니다. 다음 과제는 이번 과제의 병행성 버전을 위한 것으로 이번 과제를 제대로 완료하여야 다음 과제도 무리 없이 완료할 수 있을 것입니다.