1. Java WebScraping

a. JSON  
 - Jsoup: JSON에 따른 ajax는 callback을 확인하고 다룰 수 있지만 Document Ajax의 경우 다룰 수 없기에 HtmlUnit을 이용하여 Html코드를 받아오고 Parsing만 Jsoup사용

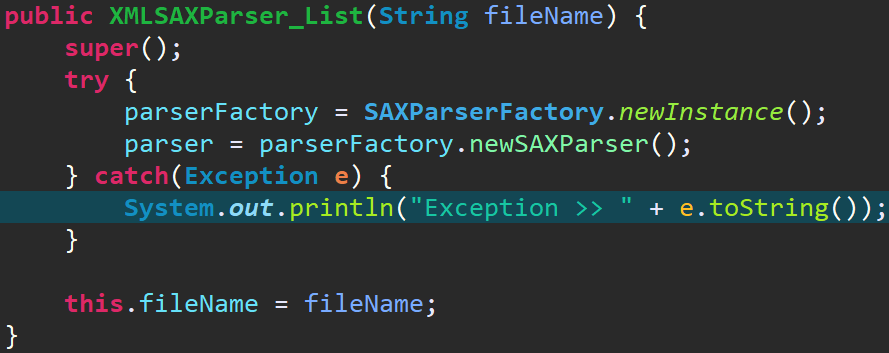
b. XML  
 🡪 DOM(Document Object Model) 방식  
 - 메모리에 모두 로드 후 파싱  
 - 노드의 검색, 수정, 구조 변경 등이 빠르고 용이  
 - 직관적이고 단순  
 - 메모리 리소스 부족 문제 발생 가능  
 🡺 문서 내용에 대한 탐색 또는 수정이 빈번한 경우 사용

🡪 SAX(Simple API for XML) 방식 **√**  
 - 단반향 Stream으로 취급하여 문서를 읽어 들이는 중 유효한 요소가 식별되면 이벤트를 외부로 전달  
 - 노드 탐색, 이동, 추가, 삭제 등의 작업이 쉽지 않음  
 - 적은 메모리 사용  
 🡺 문서의 구조가 간단하거나 동일한 요소들이 반복되는 경우 사용

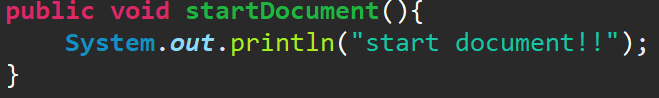
b. XML(eXtensible Markup Language)

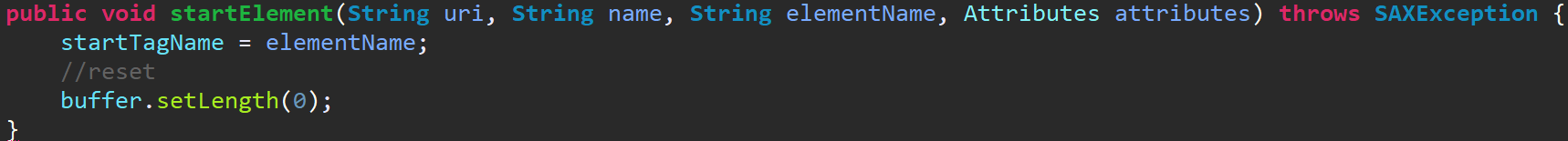
- 확장될 수 있는 표시 언어(사용자의 임의 태그 사용 가능)

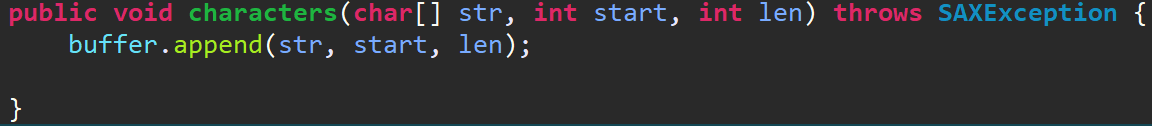
c-1. SAXParserFactory

* 어플리케이션이 SAX베이스의 Parser를 구성 및 취득해 XML 문서를 구문 분석할 수 있도록 하는 Factory API
* newInstance(): SAXParserFactory의 새로운 인스턴스를 가져옴

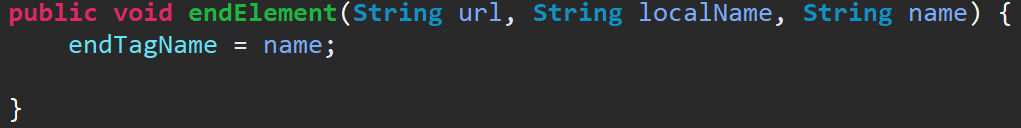
c-2. SAX(Simple API for XML)의 Method

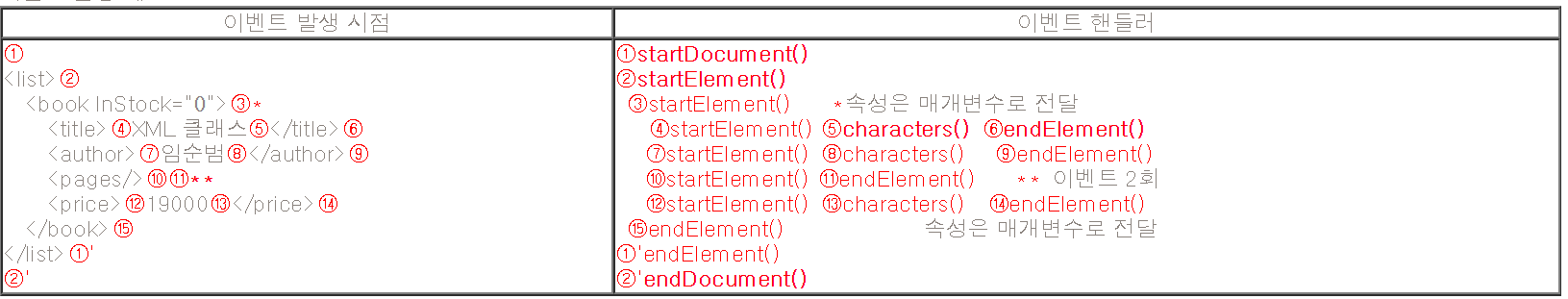
 a. startDocument: 문서의 시작에 사용

 b. startElement 시작태그를 인식했을 때 사용  
 elementName: 시작 태그 String  
 attributes: attributes.getValue(“태그명” or Index): 내용

c. characters: 시작태그와 끝 태그 사이의 내용을 인식했을 때 처리

buffer.append(str, start, len): str이란 문자 배열의 start부분부터 len만큼 buffer에 추가

 d. endDocument: 문서의 종료에 사용

 name: 끝 태그 String

참고: <http://newmkka.tistory.com/10>