Practical aspects of parsing

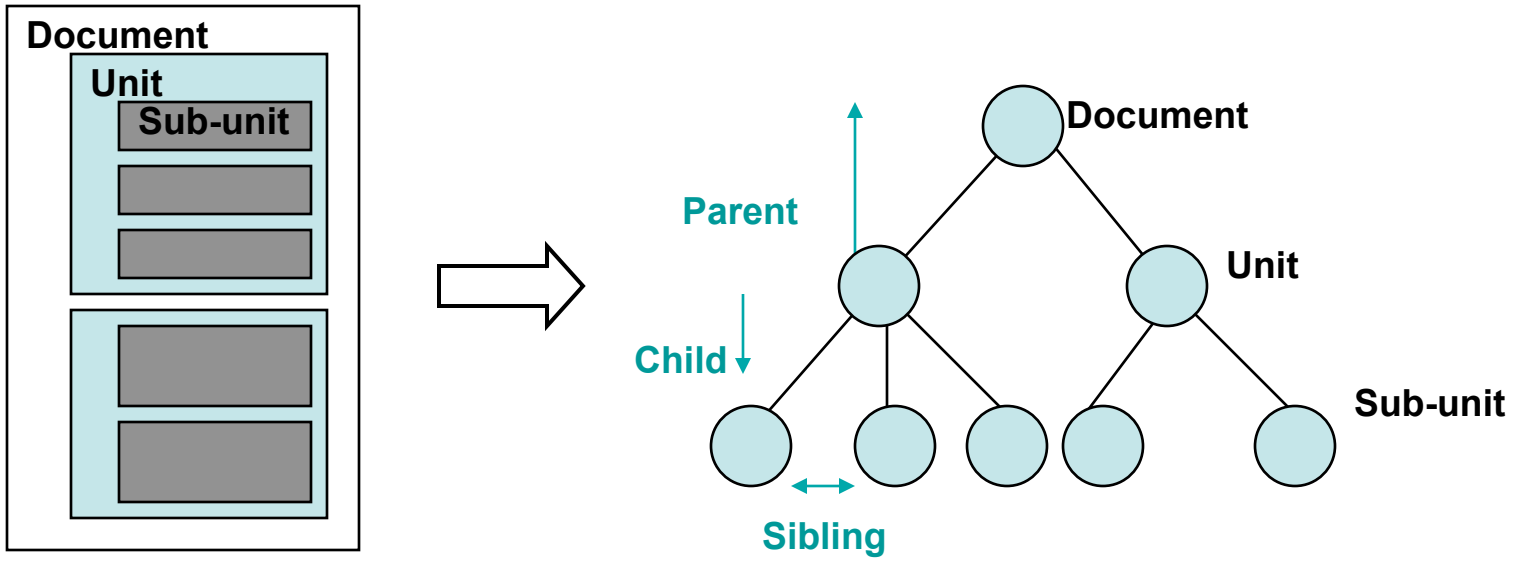
- Parser and parser generator;
- Markup languages
  - DTD, XML Schema
- XML parsing
  - DOM parser
  - SAX parser (more efficient)
  - JSOUP (more robust)
- Alternative approach to program analysis
  - doclet

DOM (Document Object Model)

- What: DOM is application programming interface (API) for processing XML documents
  - http://www.w3c.org/DOM/
- Why: unique interface. platform and language independent
- How: It defines the logical structure of documents and the way to access and manipulate it
  - With the Document Object Model, one can
    - Create an object tree
    - Navigate its structure
    - Access, add, modify, or delete elements etc

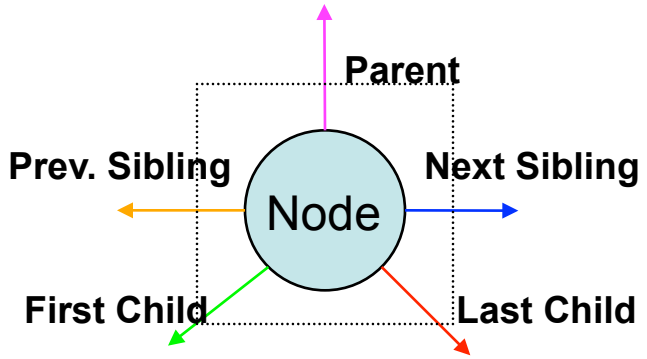
XML tree hierarchy

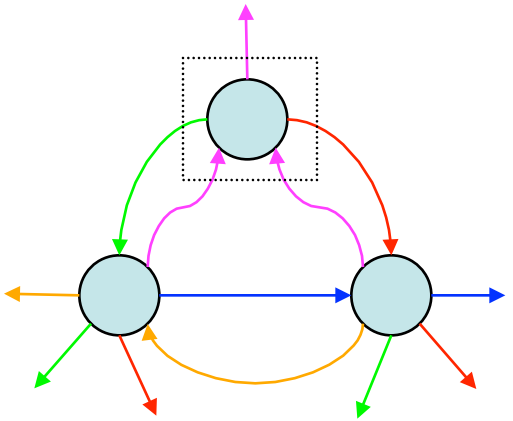
- XML can be described by a tree hierarchy



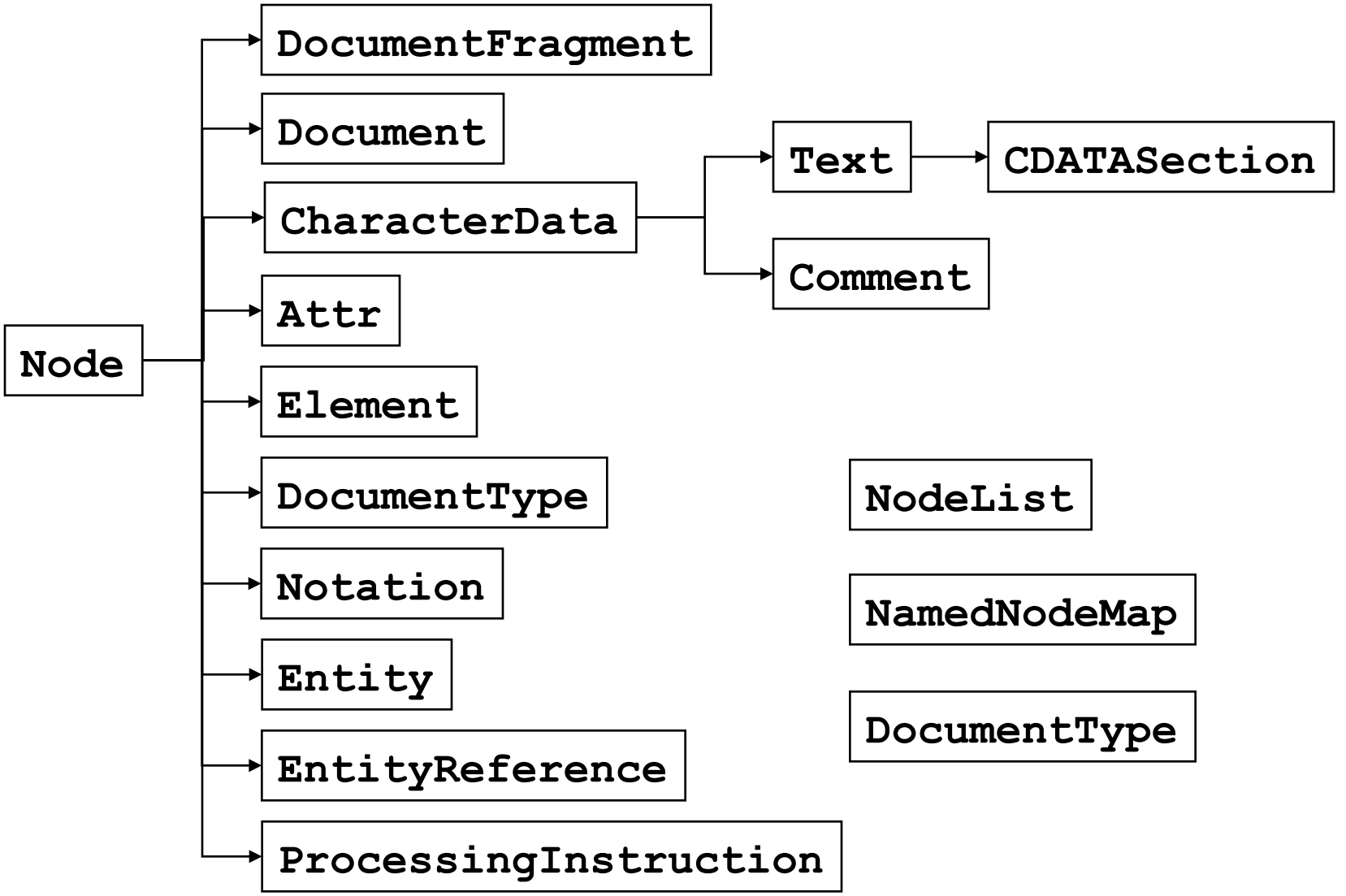
DOM tree model

- Generic tree model
  - Node
    - Type, name, value
    - Attributes
    - Parent node
    - Previous, next sibling nodes
    - First, last child nodes
  - Many other entities ***extends*** node
    - Document
    - Element
    - Attribute
    - ... ...





DOM class hierarchy



JavaDoc of DOM API

http://xml.apache.org/xerces-j/apiDocs/index.html

org.apache.xml.serialize
org.w3c.dom
org.w3c.dom.events
org.w3c.dom.html
org.w3c.dom.ranges
org.w3c.dom.traversal
org.xml.sax
org.xml.sax.ext
org.xml.sax.helpers

org.w3c.dom
Interfaces
Attr
CDATASection
CharacterData
Comment
Document
DocumentFragment
DocumentType
DOMImplementation
Element
Entity
EntityReference
NamedNodeMap
Node
NodeList
Notation
ProcessingInstruction
Text
Exceptions
DOMException

Overview Package Class Use Tree Deprecated Index Help
PREV CLASS NEXT CLASS
FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD

org.w3c.dom
Interface Document

All Known Subinterfaces:
HTMLDocument, WMLDocument

All Known Implementing Classes:
CoreDocumentImpl

public interface Document
extends Node

The Document interface represents the entire HTML or XML document. Conceptually, it is the root of the document tree, and provides the prin access to the document's data.

Since elements, text nodes, comments, processing instructions, etc. cannot exist outside the context of a Document, the Document interface als contains the factory methods needed to create these objects. The Node objects created have a ownerDocument attribute which associates them v Document within whose context they were created.

See also the Document Object Model (DOM) Level 2 Core Specification.

Fields inherited from interface org.w3c.dom.Node
ATTRIBUTE_NODE, CDATA_SECTION_NODE, COMMENT_NODE, DOCUMENT_FRAGMENT_NODE, DOCUMENT_NODE, DOCUMENT_TYPE_NODE
ELEMENT_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, NOTATION_NODE, PROCESSING_INSTRUCTION_NODE, TEXT_NODE

Method Summary
Node adoptNode(Node source)
EXPERIMENTAL! Based on the Document Object Model (DOM) Level 3 Core Working Draft of 5 June 20

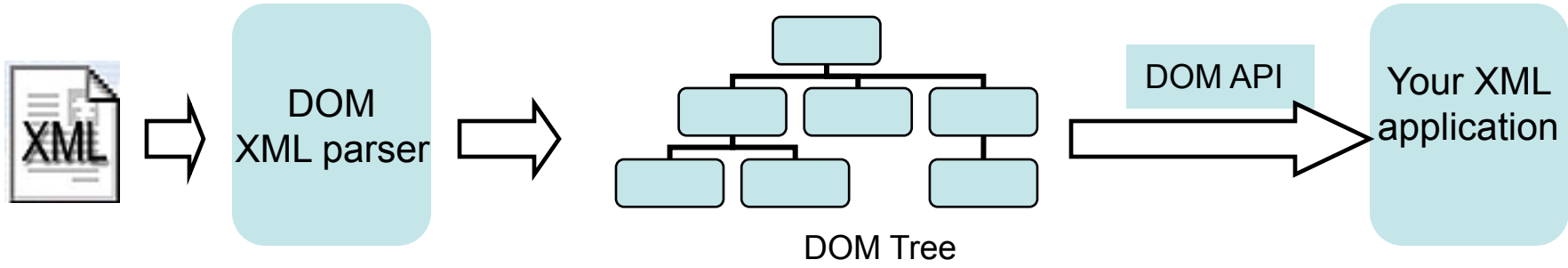
Some important DOM interfaces

- The DOM defines several Java interfaces
  - **`Node`** The base data type of the DOM
  - **`Element`** Represents element
  - **`Attr`** Represents an attribute of an element
  - **`Text`** The content of an element or attribute
  - **`Document`** Represents the entire XML document. A Document object is often referred to as a DOM tree
  - … …

Methods in Node interface

- Three categories of methods
  - Node characteristics
    - name, type, value
  - Contextual location and access to relatives
    - parents, siblings, children, ancestors, descendants
  - Node modification
    - Edit, delete, re-arrange child nodes

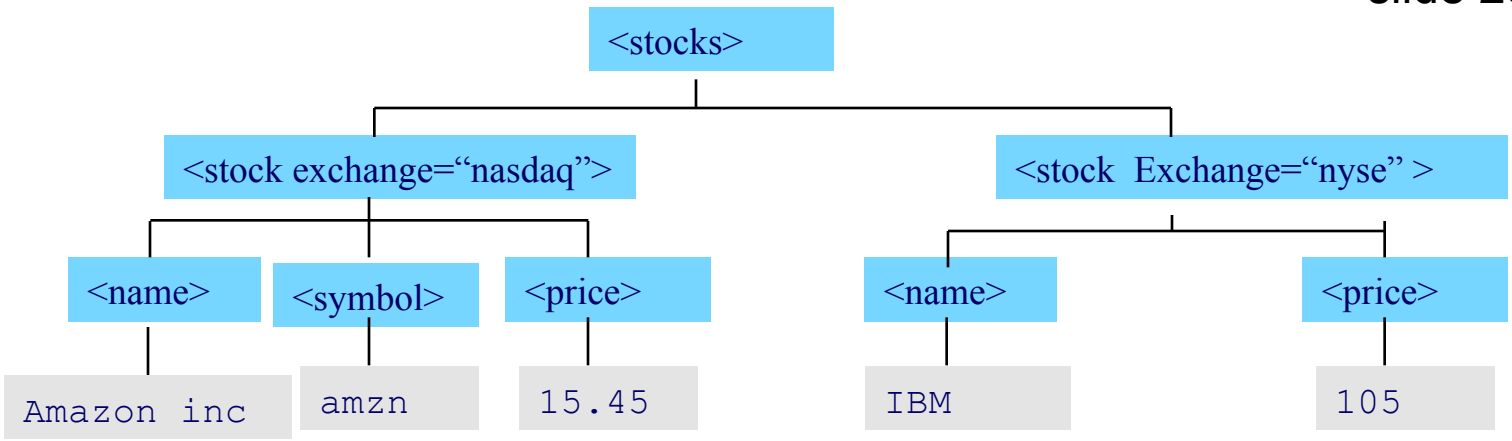
XML parser and DOM

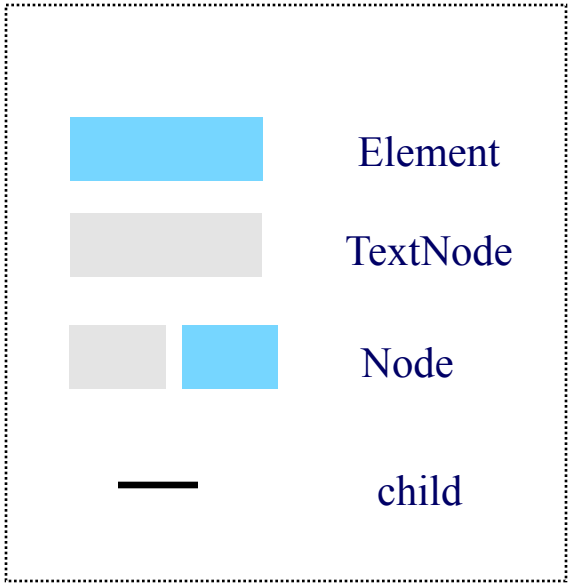


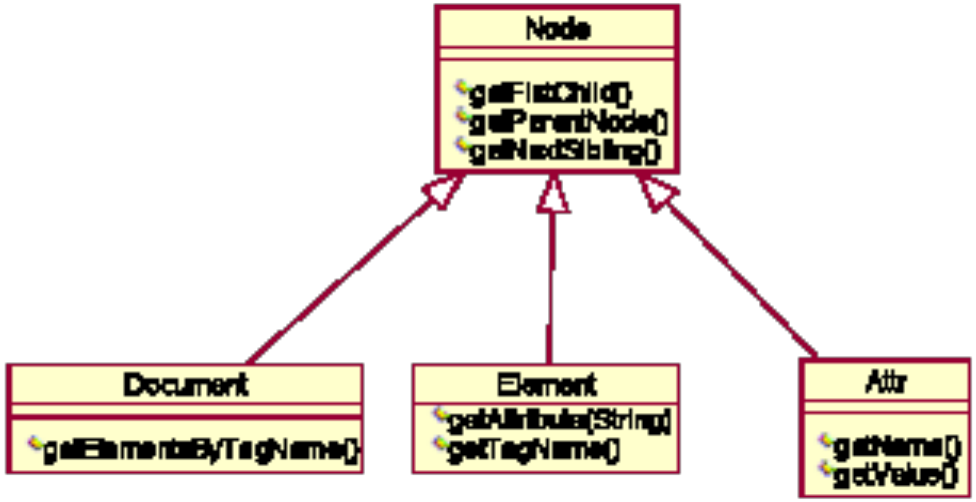
- When you parse an XML document with a DOM parser, you get back a tree structure that contains all of the elements of your document;
- DOM also provides a variety of functions you can use to examine the contents and structure of the document.

DOM tree and DOM classes

there are more nodes. refer slide 28.







Use Java to process XML

- Tasks:
  - How to construct the DOM tree from an XML text file?
  - How to get the list of stock elements?
  - How to get the attribute value of the second stock element?
- Construct the Document object:
  - Need to use an XML parser (XML4J);
  - remember to import the necessary packages;
  - The benefits of DOM: the following lines are the only difference if you use another DOM XML parser.

```
Document doc = null;
DOMParser parser = new DOMParser();
parser.parse("c:/stock.xml");
doc = parser.getDocument();
```

Get the first stock element

```
<?xml version="1.0" ?>
<stocks>
  <stock exchange="nasdaq">
    <name>amazon corp</name>
    <symbol>amzn</symbol>
    <price>16</price>
  </stock>
  <stock exchange="nyse">
    <name>IBM inc</name>
    <price>102</price>
  </stock>
</stocks>
```

```
NodeList stockList = doc.getElementsByTagName("stock");
Element stock = (Element) stockList.item(0);
System.out.println("attribute: " + stock.getAttribute("exchange"));
```

Navigate to the next sibling of the first stock element

```xml
<?xml version="1.0" ?>
<stocks>
  <stock exchange="nasdaq">
    <name>amazon corp</name>
    <symbol>amzn</symbol>
    <price>16</price>
  </stock>
  <stock exchange="nyse">
    <name>IBM inc</name>
    <price>102</price>
  </stock>
</stocks>
```

```
Element sibling = (Element)stock.getNextSibling().getNextSibling();
System.out.println("sibling attribute: " + sibling.getAttribute("exchange"));
```

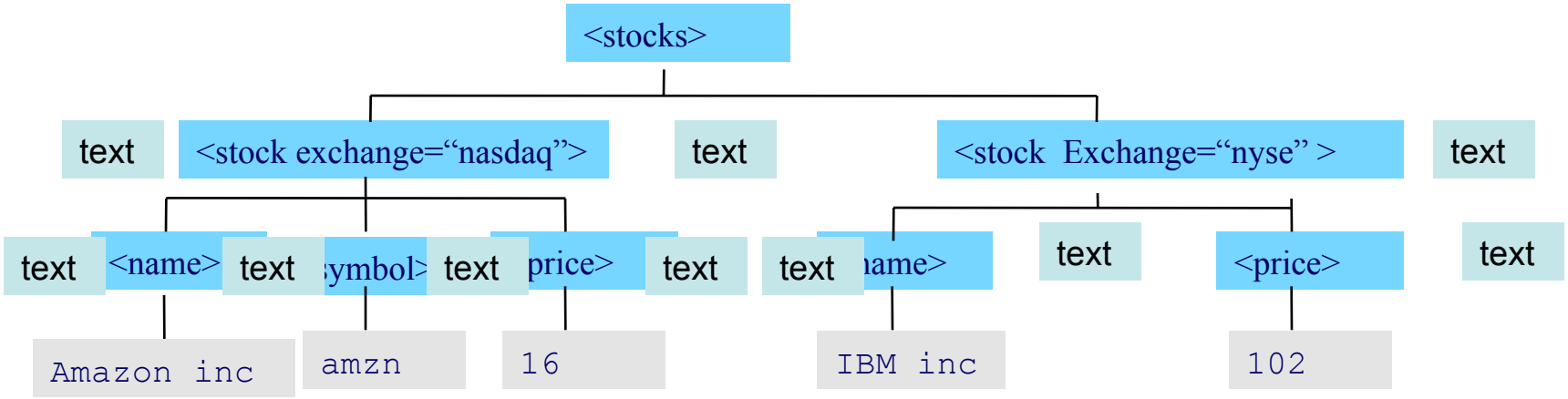
Output

```
attribute: nasdaq
sibling attribute: nyse
```

Be aware the Text object in two elements

```
<?xml version="1.0" ?>
<stocks>
  <stock exchange="nasdaq">
    <name>amazon corp</name>
    <symbol>amzn</symbol>
    <price>16</price>
  </stock>
  <stock exchange="nyse">
    <name>IBM inc</name>
    <price>102</price>
  </stock>
</stocks>
```

Question: How many children does the stocks node have?



SAX (Simple API for XML) parser

- Consider that you need to parse a large xml document
  - E.g., the entire wikipedia file, google knowledge graph
  - Giga bytes in size
  - Can not fit in memory
- DOM tree needs to be loaded into memory
- SAX parser operates on each element of XML sequentially

```
public void startElement(String uri, String localName, String qName, Attributes attributes) throws Exception {
        String name = attributes.getValue(0);

        if (qName.equals(ELE_NAME)) {
                isTheElement = true;
                sourceNode=attributes.getValue("rdf:ID");
        }
        if (isTheElement) {
                if (qName.equals("rdfs:subClassOf")) {
                        System.out. …
                }
        }
    }

public void endElement(String uri, String localName, String qName) throws SAXException {
        if (qName.equals(ELE_NAME)) {
   …

}
```

```
import org.xml.sax.*;
…
public class DBPEDIA_SAX extends DefaultHandler {

public static void main(String[] args) throws Exception {
        SAXParserFactory factory = SAXParserFactory.newInstance();
        SAXParser saxParser = factory.newSAXParser();
        XMLReader reader = saxParser.getXMLReader();
        reader.setFeature("http://apache.org/xml/features/nonvalidating/load-external-dtd",false);
        reader.setFeature("http://xml.org/sax/features/validation", false);
        saxParser.parse(new File(DBPEDIA_FILE), new DBPEDIA_SAX());
}
```

HTML parser Jsoup

- HTML is a special kind of XML
  - Why do we need yet another parser?
- Consider
  - You have an HTML document that is not a valid XML document
  - Unclosed tags
  - Nesting is not correct
  - OK to display, Will break an XML parser
- Other features specifically designed for html
  - scrape and parse HTML from a URL, file, or string
  - find and extract data, using DOM traversal or CSS selectors
  - manipulate the HTML elements, attributes, and text
  - clean user-submitted content against a safe white-list, to prevent XSS attacks
  - output tidy HTML

Jsoup example

```
Document doc = Jsoup.connect("http://en.wikipedia.org/").get();
Elements newsHeadlines = doc.select("#mp-itn b a");

Elements links = doc.select("a[href]"); // a with href
Elements pngs = doc.select("img[src$=.png]");  // img with src ending .png

Element masthead = doc.select("div.masthead").first();
  // div with class=masthead

Elements resultLinks = doc.select("h3.r > a"); // direct a after h3
```

Analyzing source code using doclet

- Task:
  - get all the methods of JDK
  - Parameter types for each method

- java.util.ArrayList
  - indexOf
    - o: java.lang.Object
  - lastIndexOf
    - o: java.lang.Object
  - Get
    - index: int
  - …
- …

Doclet example

javadoc -J-Xmx2048m -doclet ListMethodsDoclet -sourcepath source java.util

```
import com.sun.javadoc.*; …
public class ListMethodsDoclet {
    public static boolean start(RootDoc root) {
        ClassDoc[] classes = root.classes();
        for (int i = 0; i < classes.length; i++) {
            System.out.print("\n"+classes[i]);
            MethodDoc[] methods = classes[i].methods();
             for (int j = 0; j < methods.length; j++) {
                MethodDoc m = methods[j];
                    System.out.println("\t" + m.name());
                    Parameter[] parameters = m.parameters();
                    for (int k = 0; k < parameters.length; k++) {
                        Parameter p = parameters[k];
                         System.out.println("\t\t" + p.name() + ": " + p.type().qualifiedTypeName());
}}}}
        return true;
}}
```