

XML 문서 처리

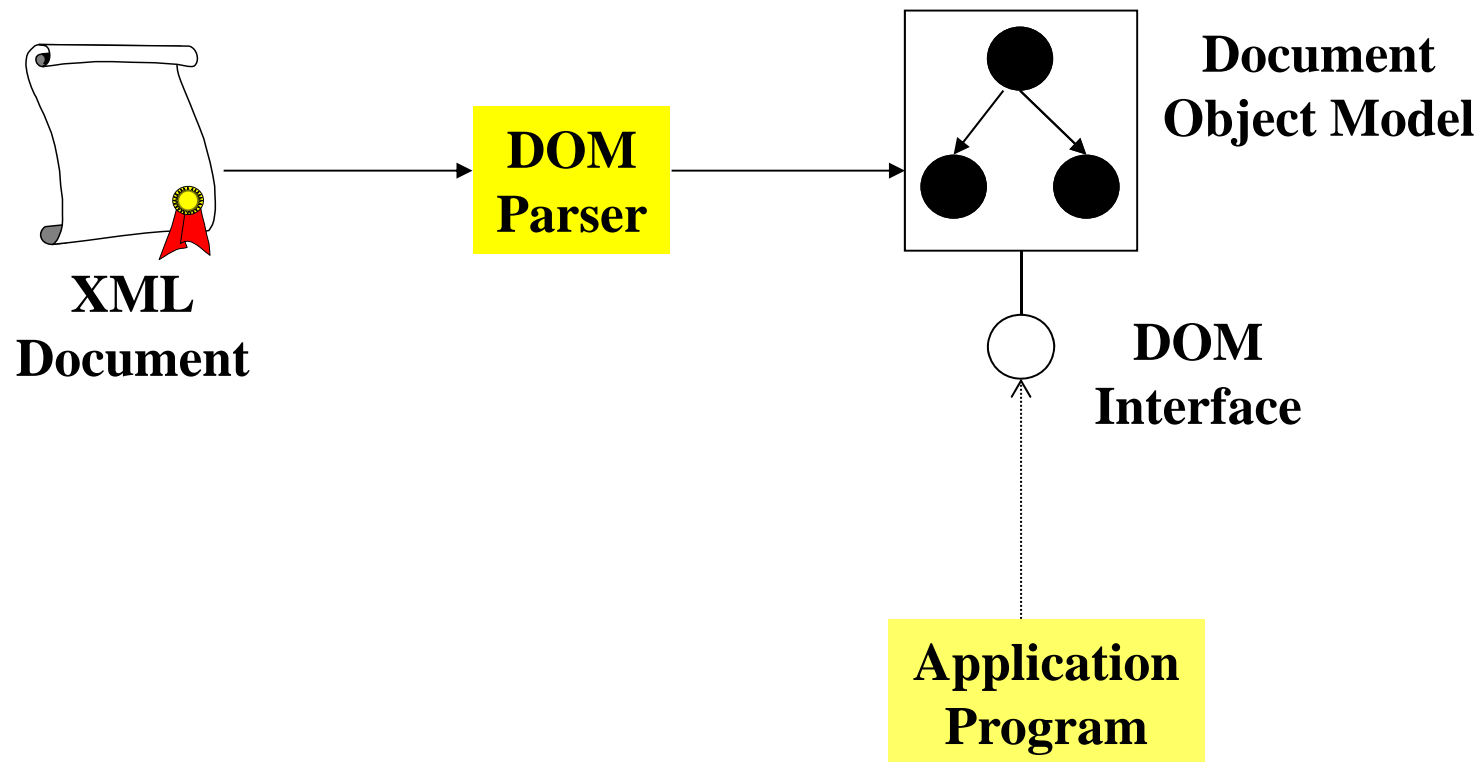
- DOM -

XML 파서

- XML Parser
 - XML 문서를 해석
 - 해석된 내용을 응용프로그램에 전달
- XML Parser의 종류
 - DOM (Document Object Model) 파서
 - 문서 해석 결과를 객체 Tree 구조로 생성
 - SAX (Simple API for XML) 파서
 - 해석하면서 동시에 Event를 발생

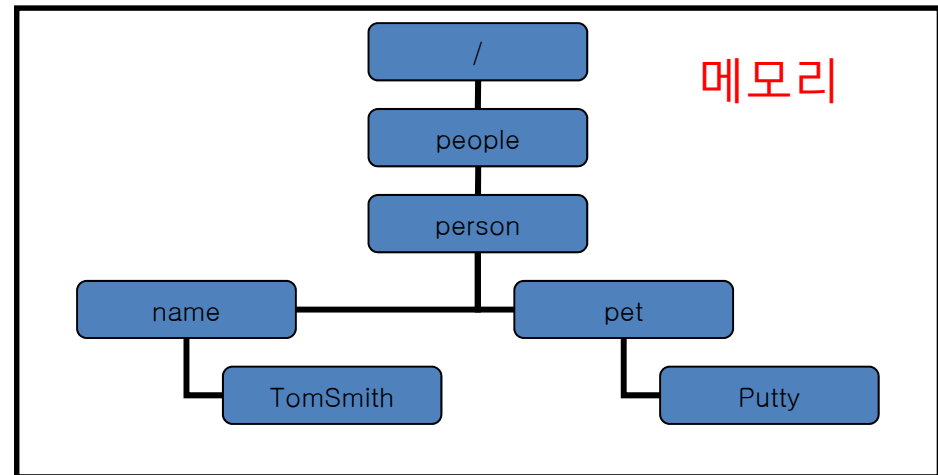
비교	DOM	SAX
노드접근	임의의 직접접근방식	순차적 접근방식
메모리 공간	전체 XML 문서에 대한 트리 구조를 저장	이벤트 처리방식으로 효율적 메모리 활용
활용분야	일반적 문서 처리	대규모의 방대한 문서처리

DOM 처리 프로세스



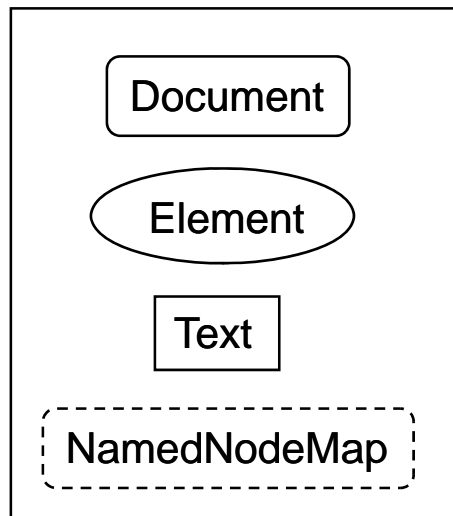
```
<people>
  <person>
    <name>TomSmith</name>
    <pet>Putty</pet>
  </person>
</people>
```

DOM
파싱

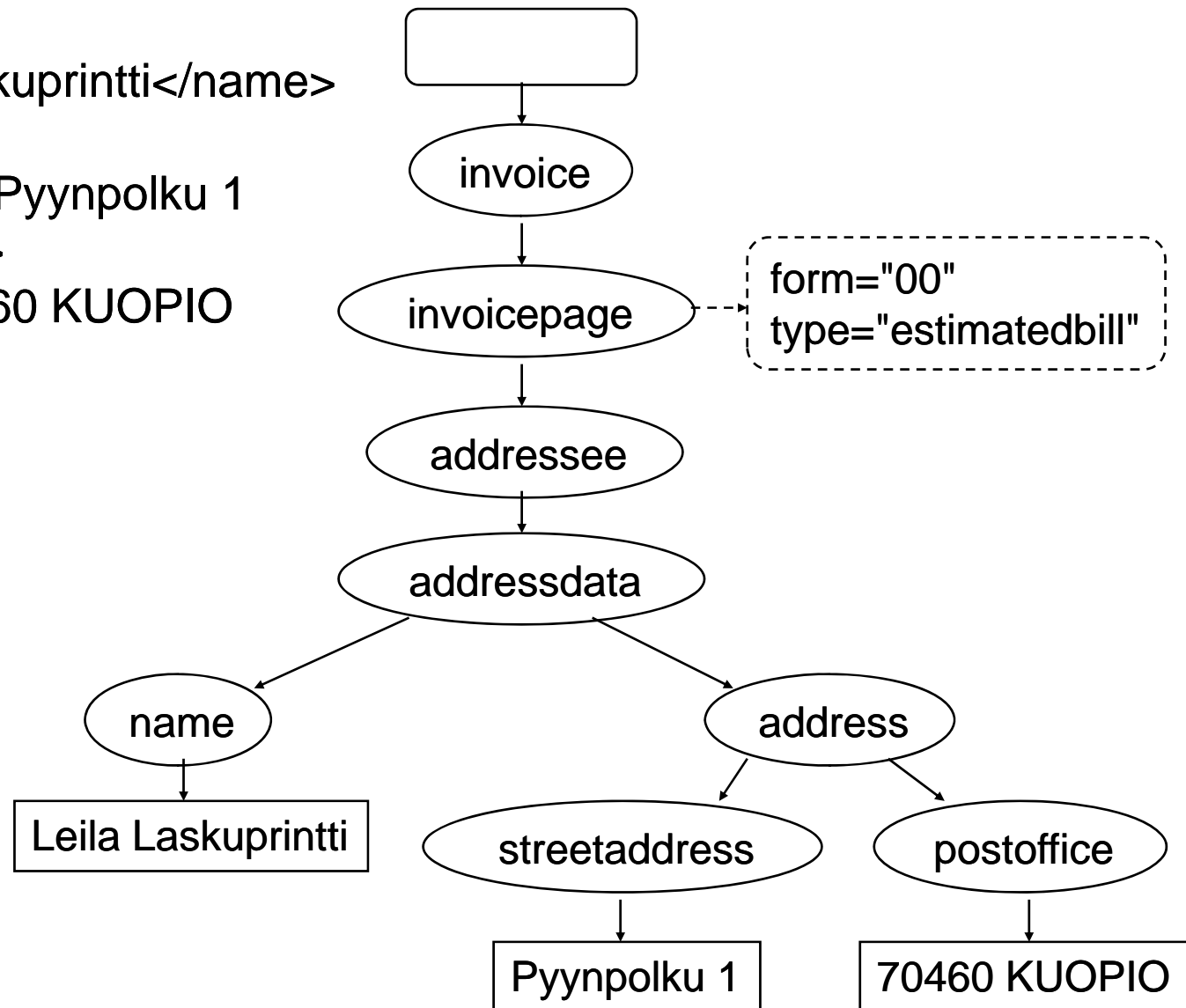


```

<invoice>
  <invoicepage form="00"
                type="estimatedbill">
    <addressee>
      <addressdata>
        <name>Leila Laskuprintti</name>
        <address>
          <streetaddress>Pyynpolku 1
          </streetaddress>
          <postoffice>70460 KUOPIO
          </postoffice>
        </address>
      </addressdata>
    </addressee> ...
  
```



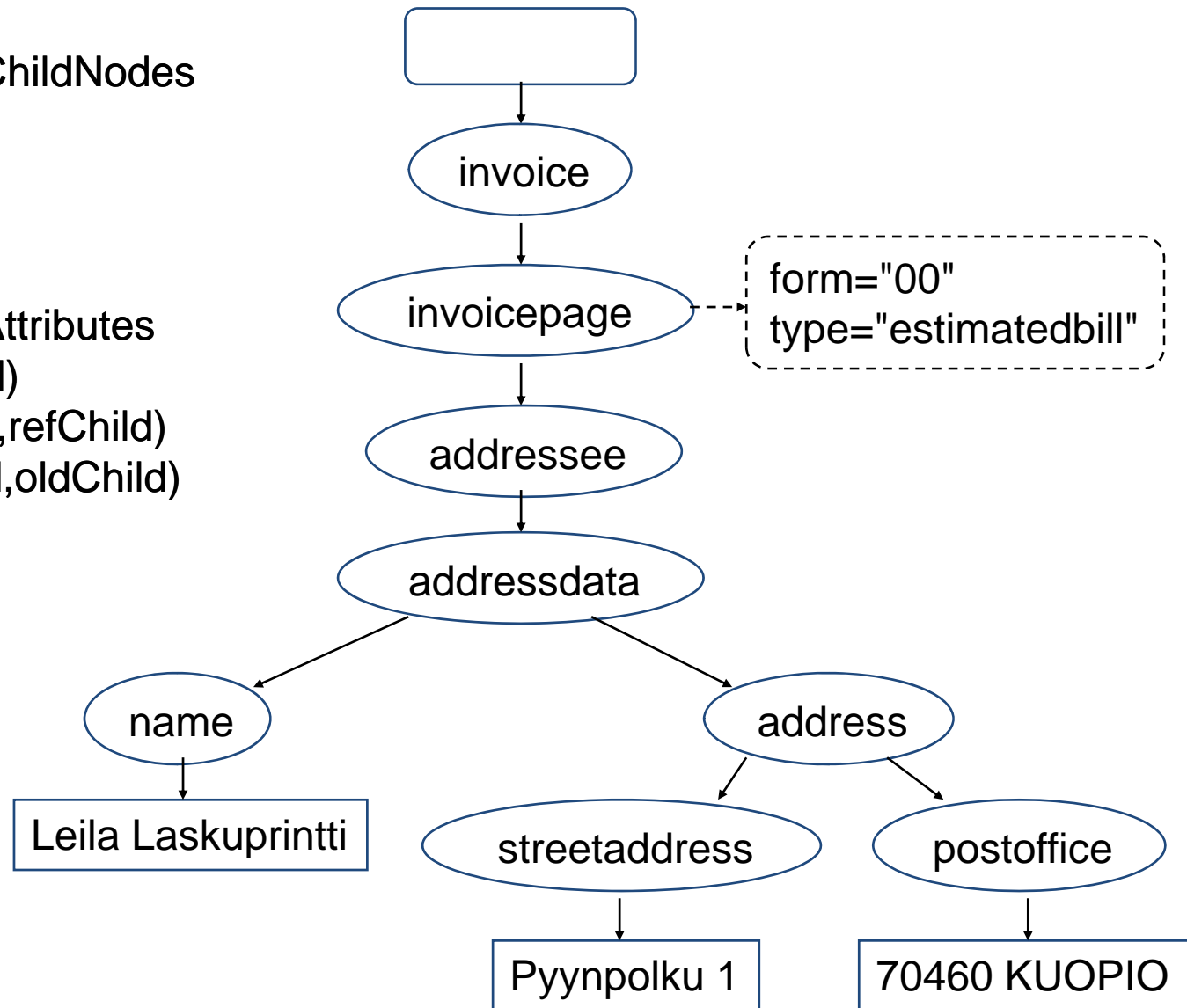
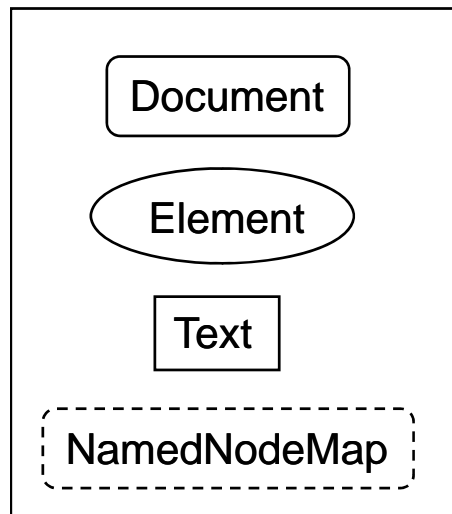
DOM structure model



Node

getNodeTypes
getNodeValue
getOwnerDocument
getParentNode
hasChildNodes getChildNodes
getFirstChild
getLastChild
getPreviousSibling
getNextSibling
hasAttributes getAttributes
appendChild(newChild)
insertBefore(newChild,refChild)
replaceChild(newChild,oldChild)
removeChild(oldChild)

DOM interfaces: **Node**

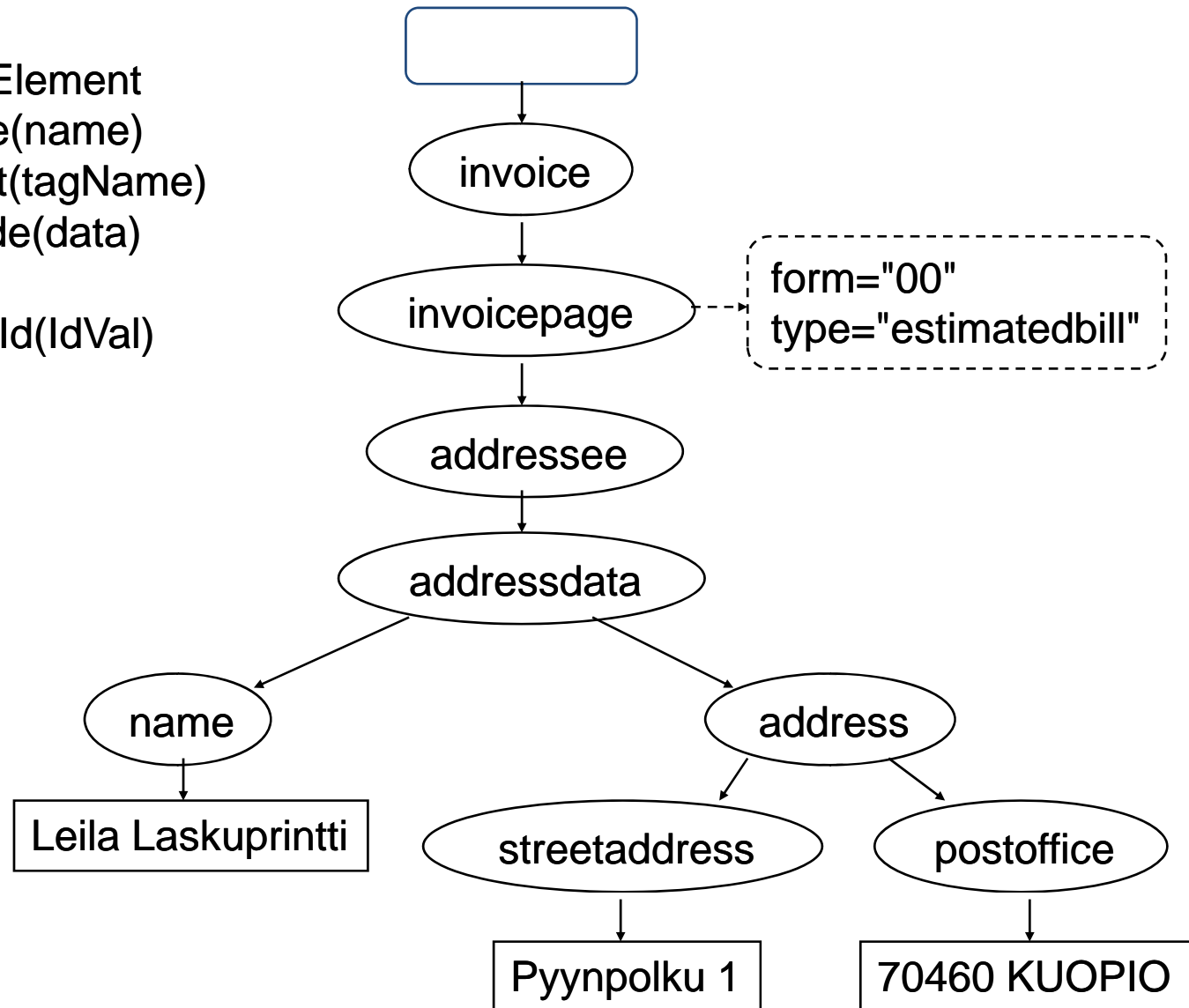
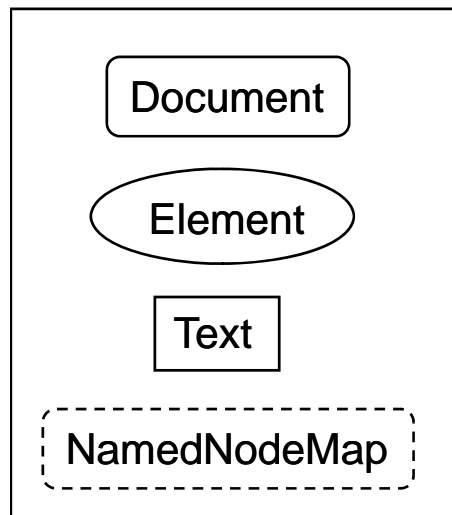


Node

Document

getDocumentElement
createAttribute(name)
createElement(tagName)
createTextNode(data)
getDocType()
getElementById(IdVal)

DOM interfaces: Document

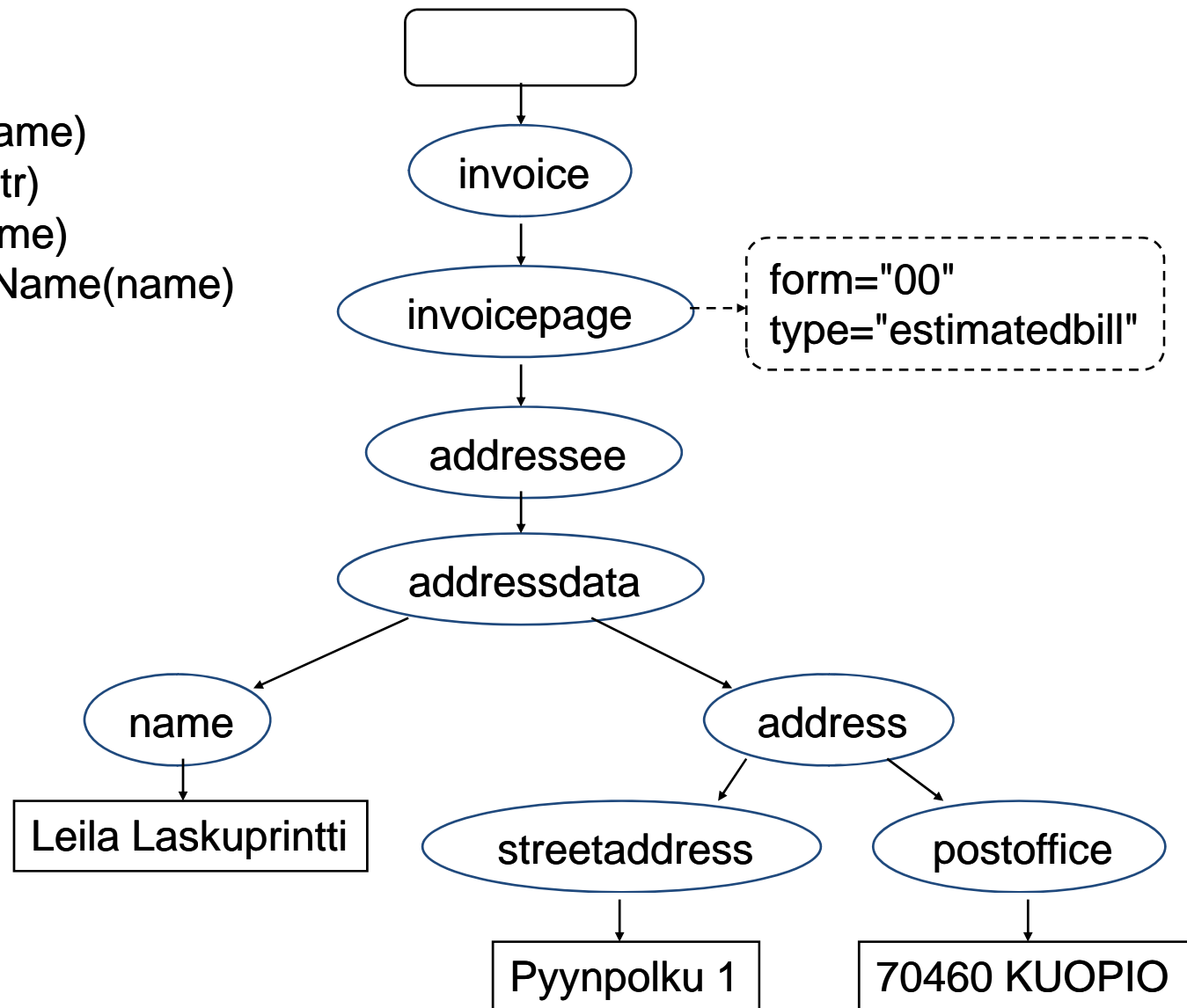
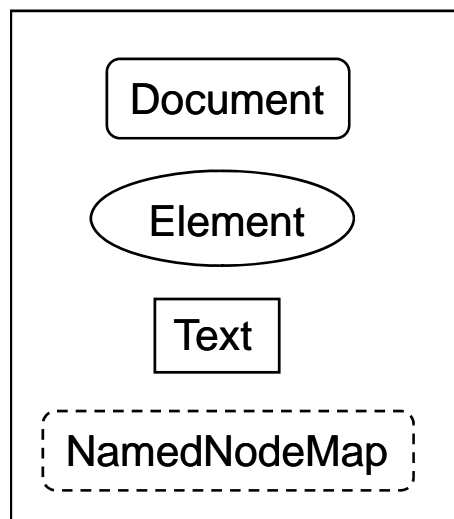


Node

Element

getTagName
getAttributeNode(name)
setAttributeNode(attr)
removeAttribute(name)
getElementsByTagName(name)
hasAttribute(name)

DOM interfaces: Element



DOM 핵심 package

- org.w3c.dom



DOM 실습예제

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
```

```
import org.w3c.dom.Element;
import org.w3c.dom.Document;
```

```
public class SimpleDOM {
```

```
    public static void main( String [] args){
```

```
        Document doc;
```

```
        try{
```

```
            //DOM 파서 생성
```

```
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
```

```
            DocumentBuilder builder = factory.newDocumentBuilder();
```

```
            //DOM 파싱
```

```
            doc = builder.parse( "dom.xml" );
```

```
            //root 엘리먼트
```

```
            Element root = doc.getDocumentElement();
```

```
            //root 엘리먼트 이름 출력
```

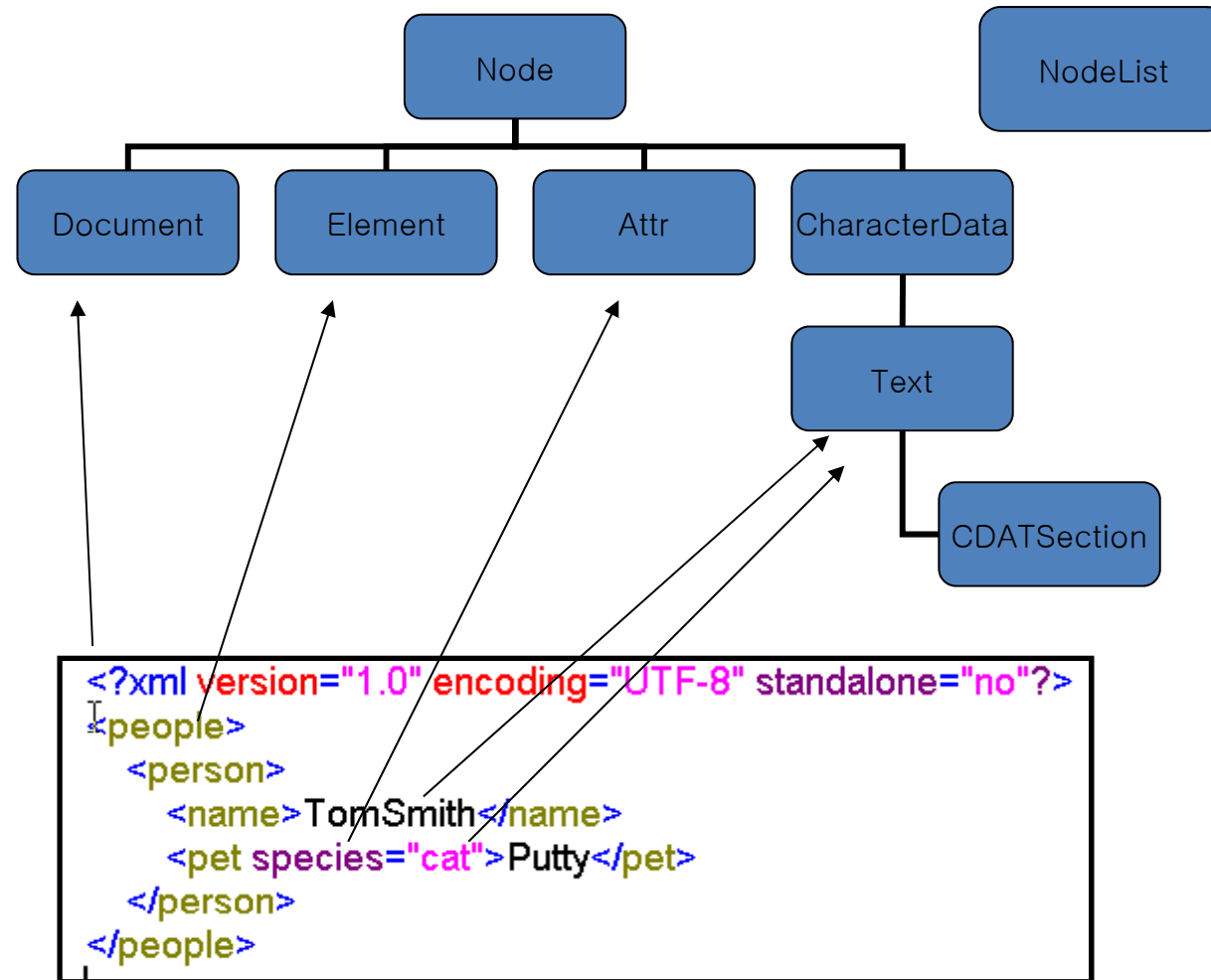
```
            System.out.println( root.getTagName() );
```

```
        }catch( Exception e){
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<people>
  <person>
    <name>TomSmith</name>
    <pet>Putty</pet>
  </person>
</people>
```

```
Problems @ Javadoc Declaration Console
<terminated> SimpleDOM [Java Application] C:\WProgr
people
```

DOM API 계층구조



DOM 파서의 parse 메소드

javax.xml.parsers javax.xml.soap javax.xml.stream javax.xml.stream.events javax.xml.stream.util javax.xml.transform javax.xml.transform.dom	abstract Document	newDocument() Obtain a new instance of a DOM Document object to build a DOM tree.
	Document	parse(File f) Parse the content of the given file as an XML document and return a Document object.
	abstract Document	parse(InputSource is) Parse the content of the given input source as an XML document and return a Document object.
	Document	parse(InputStream is) Parse the content of the given InputStream as an XML document and return a Document object.
	Document	parse(InputStream is, String systemId) Parse the content of the given InputStream as an XML document and return a Document object.
	Document	parse(String uri) Parse the content of the given URI as an XML document and return a Document object.

//웹서버 다운

```
URL url = new URL( "http://localhost/dom.xml" );  
InputStream is = url.openStream();  
Document doc = builder.parse( is );
```

"TomSmith" 데이터 찾기

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<people>
  <person>
    <name>TomSmith</name>
    <pet species="cat">Putty</pet>
  </person>
</people>
```

Methods inherited from interface org.w3c.dom.[Node](#)

[appendChild](#), [cloneNode](#), [compareDocumentPosition](#), [getAttributes](#), [getBaseURI](#), [getChildNodes](#), [getFeature](#), [getFirstChild](#), [getLastChild](#), [getLocalName](#), [getNamespaceURI](#), [getNextSibling](#), [getNodeName](#), [getNodeType](#), [getNodeValue](#), [getOwnerDocument](#), [getParentNode](#), [getPrefix](#), [getPreviousSibling](#), [getTextContent](#), [getUserData](#), [hasAttributes](#), [hasChildNodes](#), [insertBefore](#), [isDefaultNamespace](#), [isEqualNode](#), [isSameNode](#), [isSupported](#), [lookupNamespaceURI](#), [lookupPrefix](#), [normalize](#), [removeChild](#), [replaceChild](#), [setNodeValue](#), [setPrefix](#), [setTextContent](#), [setUserData](#)

공백문자 제거

- `factory.setIgnoringElementContentWhites`
`pace(true);`
 - `DocumentBuilder`는 기본으로 공백을 Text 객체로 인식
- 반드시 문서유형 선언이 필요
 - `<!DOCTYPE`

```

<?xml version="1.0" encoding="euc-kr" ?>
<!DOCTYPE people SYSTEM "_09_01.dtd" >
<people>
  <person>
    <name>Tom Smith</name>
    <pet>Putty</pet>
  </person>
</people>

```

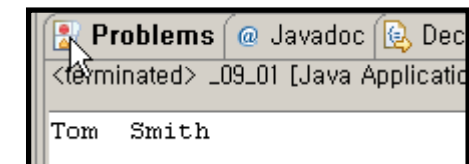
```

Document doc;
try{
    //DOM 파서 생성
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    //공백문자제거 ( space , tab , 캐리지리턴 )
    factory.setIgnoringElementContentWhitespace( true );
    DocumentBuilder builder = factory.newDocumentBuilder();

    //DOM 파싱
    doc = builder.parse( "_09_01.xml" );

    //root 엘리먼트 ( people )
    Element root = doc.getDocumentElement();
    //첫번째 child 얻기 ( person )
    Element person = ( Element ) root.getFirstChild();
    //두번째 child 얻기 ( name )
    Element name = ( Element ) person.getFirstChild();
    //"Tomsmith" 얻기 ( Text 얻기 )
    Text text = ( Text ) name.getFirstChild();
    String value = text.getData();
    System.out.println( value );
}

```



마지막 자식 객체 찾기

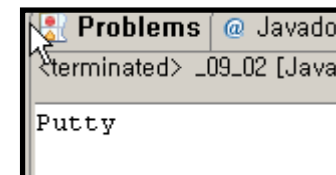
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<people>
  <person>
    <name>TomSmith</name>
    <pet species="cat">Putty</pet>
  </person>
</people>
```

```
//DOM 파싱
doc = builder.parse( "_09_02.xml" );

//root 엘리먼트 ( people )
Element root = doc.getDocumentElement();

// 마지막 child 얻기 ( person )
Element person = ( Element )root.getLastChild();

// 마지막 child 얻기 ( pet )
Element pet = ( Element )person.getLastChild();
// "cat" 얻기 ( Text 얻기 )
Text text = ( Text )pet.getFirstChild();
String value = text.getData();
System.out.println( value );
```



Problems @ Javado
<terminated> _09_02 [Java
Putty

형제 노드 찾기

//DOM 파싱

```
doc = builder.parse( "_09_03.xml" );
```

//root 엘리먼트 (people)

```
Element root = doc.getDocumentElement();
```

// 첫번째 child 얻기 (person)

```
Element person = ( Element )root.getFirstChild();
```

// 두번째 child 얻기 (name)

```
Element name = ( Element )person.getFirstChild();
```

// 형제노드 찾기 (pet)

```
Element pet = ( Element )name.getNextSibling();
```

```
Text text = ( Text )pet.getFirstChild();
```

```
String value = text.getData();
```

```
System.out.println( value );
```

```
<?xml version="1.0" encoding="euc-kr" ?>
<!DOCTYPE people SYSTEM "_09_03.dtd" >
<people>
  <person>
    <name>Tom Smith</name>
    <pet>Putty</pet>
  </person>
</people>
```



속성값 얻기

- `getAttribute("속성명");`

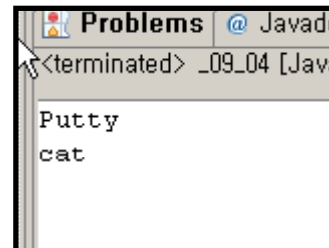
```
//root 엘리먼트 ( people )
Element root = doc.getDocumentElement();
I
//첫번째 child 얻기 ( person )
Element person = ( Element )root.getFirstChild();

//두번째 child 얻기 ( name )
Element name = ( Element )person.getFirstChild();
// 형제노드 찾기 ( pet )
Element pet = ( Element )name.getNextSibling();

Text text = ( Text )pet.getFirstChild();
String value = text.getData();
System.out.println( value );
System.out.println( pet.getAttribute( "species" ) );

}catch( Exception e){
    e.printStackTrace();
}
```

```
<?xml version="1.0" encoding="euc-kr" ?>
<!DOCTYPE people SYSTEM "_09_04.dtd" >
<people>
  <person>
    <name>Tom Smith</name>
    I <pet species="cat">Putty</pet>
  </person>
</people>
```



Problems @ Javadd

<terminated> _09_04 [Java

Putty
cat

동일 이름의 엘리먼트 객체 찾기

- `elm.getElementsByTagName("tag명")`
 - `NodeList`를 return

```
//DOM 파싱
doc = builder.parse( "_09_05.xml" );

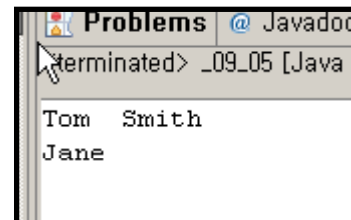
//root 엘리먼트 ( people)
Element root = doc.getDocumentElement();

NodeList list = root.getElementsByTagName( "name" );
int count = list.getLength();

for ( int i = 0 ; i < count ; i++ ){
    Element name = ( Element ) list.item( i );

    Text text = ( Text ) name.getFirstChild();
    String value = text.getData();
    System.out.println( value );
}
```

```
<?xml version="1.0" encoding="euc-kr" ?>
<!DOCTYPE people SYSTEM "_09_05.dtd" >
<people>
  <person>
    <name>Tom Smith</name>
    <pet species="cat">Putty</pet>
  </person>
  <person>
    <name>Jane</name>
    <pet species="dog">Fuffy</pet>
  </person>
</people>
```



Problems @ Javadoc
terminated> _09_05 [Java A
Tom Smith
Jane

DOM 객체 수정

- Text 객체 수정
 - `text.setData("변경Data");`
- 속성 객체 수정
 - `elm.setAttribute("속성명", "값");`
- 주의
 - DOM 객체를 수정한다고 해서 실제 XML 파일이 변경되는 것은 아님
 - 단지 메모리에 저장된 값이 변경
 - 변경된 값을 XML 파일까지 적용을 시키려면 저장 작업이 필요

```

<?xml version="1.0" encoding="euc-kr" ?>
<!DOCTYPE people SYSTEM "_09_06.dtd" >
<people>
  <person>
    <name>Tom Smith</name>
    <pet species="cat">Putty</pet>
  </person>
</people>

```

```

DocumentBuilder builder = factory.newDocumentBuilder();

//DOM 파싱
doc = builder.parse( "_09_06.xml" );

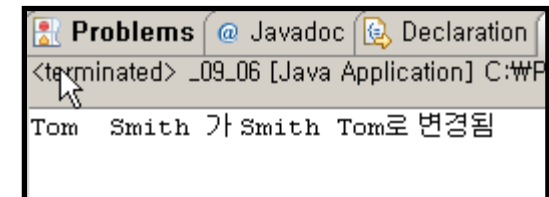
//root 엘리먼트 ( people)
Element root = doc.getDocumentElement();

Element person = ( Element )root.getFirstChild();
Element name = ( Element ) person.getFirstChild();

Text text = ( Text )name.getFirstChild();
String value = text.getData();
text.setData( "Smith Tom" );
String changeValue = text.getData();
System.out.println( value + " 가 " + changeValue + "로 변경됨");

catch( Exception e)

```



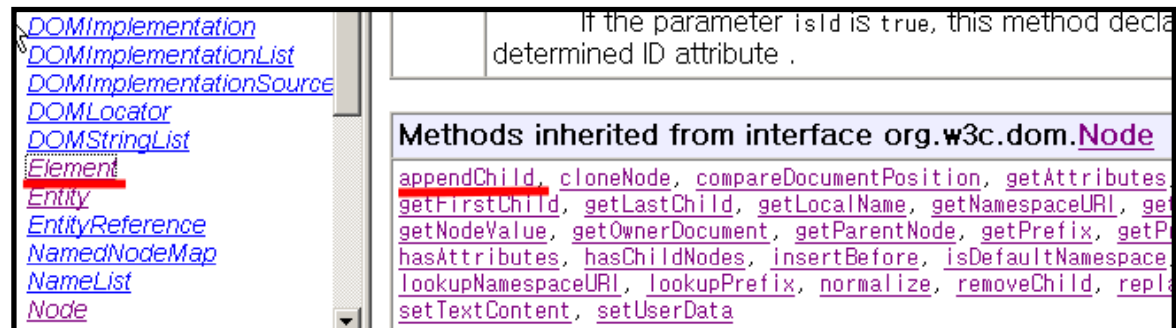
DOM 객체 추가

- 객체를 생성
 - doc.createElement("엘리먼트명");
 - doc.createTextNode("데이터");

CDATASection	CDATASection	createCDATASection (String data) Creates a CDATASection node with the specified data.
CharacterData		
Comment	Comment	createComment (String data) Creates a Comment node given the specified data.
Document		
DocumentFragment	DocumentFragment	createDocumentFragment () Creates an empty DocumentFragment node.
DocumentType		
DOMConfiguration		
DOMError		
DOMErrorHandler		
DOMImplementation	Element	createElement (String tagName) Creates an element of the type specified by tagName.
DOMImplementationList		

DOM 객체 추가

- 객체 붙이기
 - `elm.appendChild(childElement);`
- 속성 객체일 경우
 - `setAttribute("속성명", "값");`
 - 속성이 있으면 수정, 없으면 생성




```

//root 엘리먼트 ( people)
Element root = doc.getDocumentElement();
// hobby 엘리먼트 추가
Element hobby = doc.createElement( "hobby");
// hobby 텍스트 추가
Text hText = doc.createTextNode( "FootBall" );
hobby.appendChild( hText );

Element person = ( Element )root.getFirstChild();

person.appendChild( hobby );

Element name = ( Element ) person.getFirstChild();

Element pet = ( Element )name.getNextSibling();
Element nHobby = ( Element ) pet.getNextSibling();

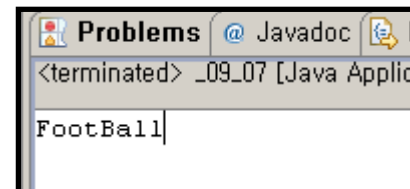
Text nText = ( Text ) nHobby.getFirstChild();
System.out.println( nText.getData() );

```

```

<?xml version="1.0" encoding="euc-kr" ?>
<!DOCTYPE people SYSTEM "_09_07.dtd" >
<people>
  <person>
    <name>Tom Smith</name>
    <pet species="cat">Putty</pet>
  </person>
</people>

```



DOM 객체 삭제

- `elm.removeChild(childElement);`
 - 부모 엘리먼트에서 자식 엘리먼트를 삭제

```
//DOM 파싱
doc = builder.parse( "_09_08.xml" );

//root 엘리먼트 ( people)
Element root = doc.getDocumentElement();
//name 엘리먼트 제거
Element person = (Element)root.getFirstChild();

root.removeChild( person );
Element name2 = (Element)person.getFirstChild();

Element person2 = ( Element ) root.getFirstChild();
Element name = (Element)person2.getFirstChild();
Text text = (Text)name.getFirstChild();
System.out.println( text.getData() );
( Exception e){
printStackTrace();
```

```
<?xml version="1.0" encoding="euc-kr" ?>
<!DOCTYPE people SYSTEM "_09_08.dtd" >
<people>
  <person>
    <name>Tom Smith</name>
    <pet species="cat">Putty</pet>
  </person>
  <person>
    <name>Jane</name>
    <pet species="dog">Fuffy</pet>
  </person>
</people>
```

Jane