Ganesha

OM SAIRAM

B.v. chalam bollapragada

9949887496

8500001926

XML

- ⟹ XML
  - ⟹ DTD
  - ⟹ parses
    - ⟹ DOM
    - ⟹ Sax
    - ⟹ Stax
- ⟹ jaxp
- ⟹ jaxb
- ⟹ XSL
- ⟹ XSU
- ⟹ Xpath ⟹ using dtd all will do,
- ⟹ Xml Schema ⟹ using Schema above tasks will do.

23/8/12

XML

XML (Extensible Markup Language)

=) XML is a metamarkup language, that includes some standards for implementing markup language.

=) For implementing Markup languages, some standards are required, and these are grouped under the name XML.

Markup Language:-

=) provides a descriptive approach to describe the content.

=) It is a language, that consists of constructs like tags/elements /attributes to describe the content for a specific (or) generic purpose.

Q) What is the advantage of Markup language?

Ans => For describing the content in a meaning full manner

=) easy to understand and implement

=) less error prone

=) for persisting temporary as well as permanent data.

IBM has introduced Markup language with the name GML then call SGML (Standard General Markup language), which was later taken over by W3C

Note:- HTML is derived for SGML.

Using SGML standards and implementing the ML's are not recommended for small projects.

=) So a collection of SGML standards and with some Additional features XML was introduced.

24/8/12 =) XML is meta markup language

⇒ ① purpose is implementing markup language's

Q) why markuplanguages?

=) Since a single ML doesn't meet all the requirement of the enterprise world

like chemical Expressions,

Mathematical Expressions... for describing this type of data we have XML ML's like CML, MML....

=) The benfit we get using xml is

a) Acts like MiniDataBase

b) Less Error Prone

c) It is extensible

d) Transfer the data in the network.

=) XML is Meta ML, providing Standards to implement ML'S.

Xml

---- provides standards

--- for Implementing XML ML'S

---- MathML

---- CML

---- WML

---- XML ML'S Standards, we prepare xml Documents.

=) As xml is a language, It Consist of a 'Specification' and an 'Implementation'.

java => document java C => Implementation.

XML are like Database, R Database, CRUD opera [persistance Data]

=> Preparing XML ML Specification

Including Declaring ==> ELEMENTS, ATTRIBUTES, ENTITIES AND NOTATIONS.

=> To (declare the above), we can use DTD or XML Schema to create xml document

Preparing the Implementation :

=> To put the xml document into Action we use implementation's parsers, for reading, writing data from the xml document.

Now we Commence preparing xml specification, for which we use DTD or XML Schema, Commencing with DTD.

Q) What is DTD (Document Type Defination)?

=> It is a part of XML ML Specification.

=> It provides Standards for declaring ELEMENTS, ATTRIBUTES ENTITIES AND NOTATIONS.

25/8/12

ELEMENTS

Element :- element describes about the enclosed Content.

Q) What are the different types of elements?

Ans

a) ChildOnly
b) TextOnly
c) Mixed
d) Empty
e) Any elements.

Child Only Elements :-

Child only Elements are the elements whose Content is, Some Other Elements.

Eg:- <servlet> ⟶ servlet is a ChildOnly element [i.e contains other elements like. <servlet-name> <servlet-class>]

```
<servlet-name> abc </servlet-name>
<servlet-class> MyServlet </servlet-class>
</servlet>
```

Sytanx for declaring ChildOnly elements, in the DTD Document

```
<! ELEMENT rootElementName (List of child elements) >
```

Let's take example:-

Xml document.

```
<Cinema>
  <cinema-name> Shiva </cinema-name>
  <cinema-hero> Nag </cinema-hero>
</cinema>
```

Dtd document

```
<! ELEMENT Cinema (cinema-name, Cinema-hero) >
```

Note

Xml Specifiers

- (*) ⟹ 0 (or) more times
- (+) ⟹ 1 (or) more times
- (?) ⟹ 0 (or) more times.

} in <web-app> occurance in <web-app>

Note:- modifer Static, public, volatie, privates, strictfp, Synchronization

for eg:- in the xml document if we have. XML-Document

```
<cinemas>
<cinema>
<cinema-name> Shiva </cinema-name>
<cinema-hero> Nag </cinema-hero>
</cinema>
<cinema>
<cinema-name> Sarkar </cinema-name>
<cinema-hero> Amitab </cinema-hero>
</cinema>
</cinemas>
```

Q) How can we this be described in the DTD document?

from our above learning

we write as

```
<!ELEMENT Cinemas (cinema, cinema)>
```

=) But we specify the occurance we have the following specifiers.

* ⟶ specifies that the occurance of the element is for 0 or more times

+ ⟹ specifies that the occurance of the element is for 1 or more times.

? ⟹ specifies that the occurance of the element is for 0 or 1 time.

Now we can correct the above as.

```
<!ELEMENT Cinemas (cinema*) >
```

⇒ If the requirement is like

web

Xml document

```
<cinemas>
<cinema>
<cinema-name> aaa </cinema-name>.
<cinema-hero> Nag </cinema-hero>
</cinema>

<cinema>
<cinema-name> ba </cinema-name>
<cinema-heroin> Tamanna </cinema-heroin>
</cinema>
</cinemas>
```

DTD Document :-

```
<! ELEMENTS Cinemas(cinema-name(cinema-hero | cinema-heroin))>
```

⇒ To describe the above in the DTD, we use pipe Symbol i.e |

Point to Note :- The element names can have numbers, but it shouldn't commence with a number, and it doesn't accept, space character, and xml markup character like <, >, !

b) Text Only Elements :

⇒ It is a datatype

⇒ Text Only Elements describes the content as text. So in the DTD Document to describe the Text Content we use

pcDATA (parsed characterdata)

Syntax :- for TextOnly elements in the DTD is

<! ELEMENT elementName (#pcDATA) >

for eg :

<Cinema>

<cinema-name> Sarkar </cinema-name>

</cinema>

=) In the above Cinema-name is TextOnly Element

=) In the DTD document we decribe this as

<!ELEMENT Cinema-name (#pcDATA) >

c) Mixed Elements :-

An Element whose Content is text / childonly / (text and Child only) / empty, Can be refered as MixedElement.

=) for this type of Elements we have the following Syntax.

<! ELEMENT elementName (#pcDATA / List of child elements Seperated by /)* >

for eg :

xml

```
<cinema>
<hero/>                    (text)
<heroin> xyz </heroin>
<villan> abc </villan>     } (text and child only)
<Commedian></Commedian>    (empty)
</cinema>
```

<! ELEMENT Cinema (#pcDATA / hero / heroin / villan / commedian)* >

d) Empty Elements :-

Empty whose Content is Empty are refered as Empty Elements.

for eg:

Syntax :- <!Element elementName EMPTY>

<!ELEMENT br EMPTY>

e) ANY Elements :-

⇒ Any elements describes, that any element can have any Content.

⇒ The use of these elements is not recommended.

⇒ we can find them used in semi-finished projects.

Syntax :- <!ELEMENT elementName ANY>

Note:- Content Elements
describe Attribute
Reusability Entities

28/8/12

Attributes :-

⇒ An attribute is a part of xml markup language Specification, or we can also say a Construct, into XML ML Specification.

Attributes are used to provide an Extra or Additional information to the Content Enclosed with in an Element.

Xml docum:- <cinema language="hindi">

<cinema-name> Sholay </cinema-name>

</cinema>

=) An attribute Consists of a name and a value.

=) The name and value are seperated using = character

=) The attribute values should be single or double quoted.

=) An element Can have multiple attributes, But the attribute name should not be duplicated.

=) The Element name and the attribute name should be Seperated using a 'space' characters.

=) Overall we say Attribute is a dependent Construct, and it resides with in the Element.

=) The Syntax To declare an Attribute in the DTD document as follows

<!ATTLIST elementName attributeName attributeSpecifier Type

attributeSpecifier. [defaultValue] >

<!ATTLIST cinema language ? ? [?]>

cinema → elementName

language → attributeName

? → attributeType.

? → attributeSpecifier

[?] → defaultValue

Types of Attributes :-

a) CDATA [Character Data] :-

=> If this Type is Applied to an attribute, The attribute value Can be alphanumeric, i.e can Contain number, alphabates and Can even Start with number, we Can Space as a delimeter.

Note :- Mark up language characters are restricted like <, >

b) NMTOKEN (Name Token) :-

=> This type is Similar To CDATA without Space character, and attribute value Should n't Commence with a Number.

< book author = "Santosh" >

c) NMTOKENS :-

=> This type takes one or more NMTOKEN types, where Space as delimeter.

< book author = "Santosh kumar" >

< book author = "Santosh 1kumar" > invalid

because Commencing with number

=> NMTOKEN, NMTOKENS not Commence with number

Only CDATA has the availability with "number" Commence

=> But In NMTokens have authority To us Space, by In NMToken has not authority of use space

d) <u>ID</u> :-

⇒ Accepts Unique Values (Similar PK)

⇒ ID ~~values~~ types can have numbers but they should not commence with a number.

Eg :- <emp eid = "e101">

e) <u>IDREF</u> :-

⇒ IDREF types takes one ID type attributes values (Similiar to FK)

for eg :-

<!ATTLIST emp eid ID #REQUIRED>
<!ATTLIST emp mgid IDREF #IMPLIED>

f) <u>IDREFS</u> :-

This type accepts one or more IDREF values, where Space is the delimeter.

g) <u>ENUM</u> :-

Describes list of values for this type we have the following Syntax.

<!ATTLIST elementName attributeName (list of values Seperated by '|') "defaultvalue">

<!ATTLIST cinema talk (hit | superhit | flop) "superhit">

<!ATTLIST emp working (yes | NO | retaried) "yes">
↓
VOI

h) ENTITY :- entity describes Unparsed entity

i) ENTITIES :-

describes one or more Unparsed entities.

(h and i we are going to learn in the next Construct)

Specifiers :-

REQUIRED :- Specifies that the attribute is mandatory in the xml document.

IMPLIED :- Specifies that the attribute is Optional

FIXED :- Specifies that the attribute is optional, if this Specified is applied, then we have to Specify the default value in the attribute Syntax.

ENTITIES

=) Entities :- Entities are used to Resue the Content.

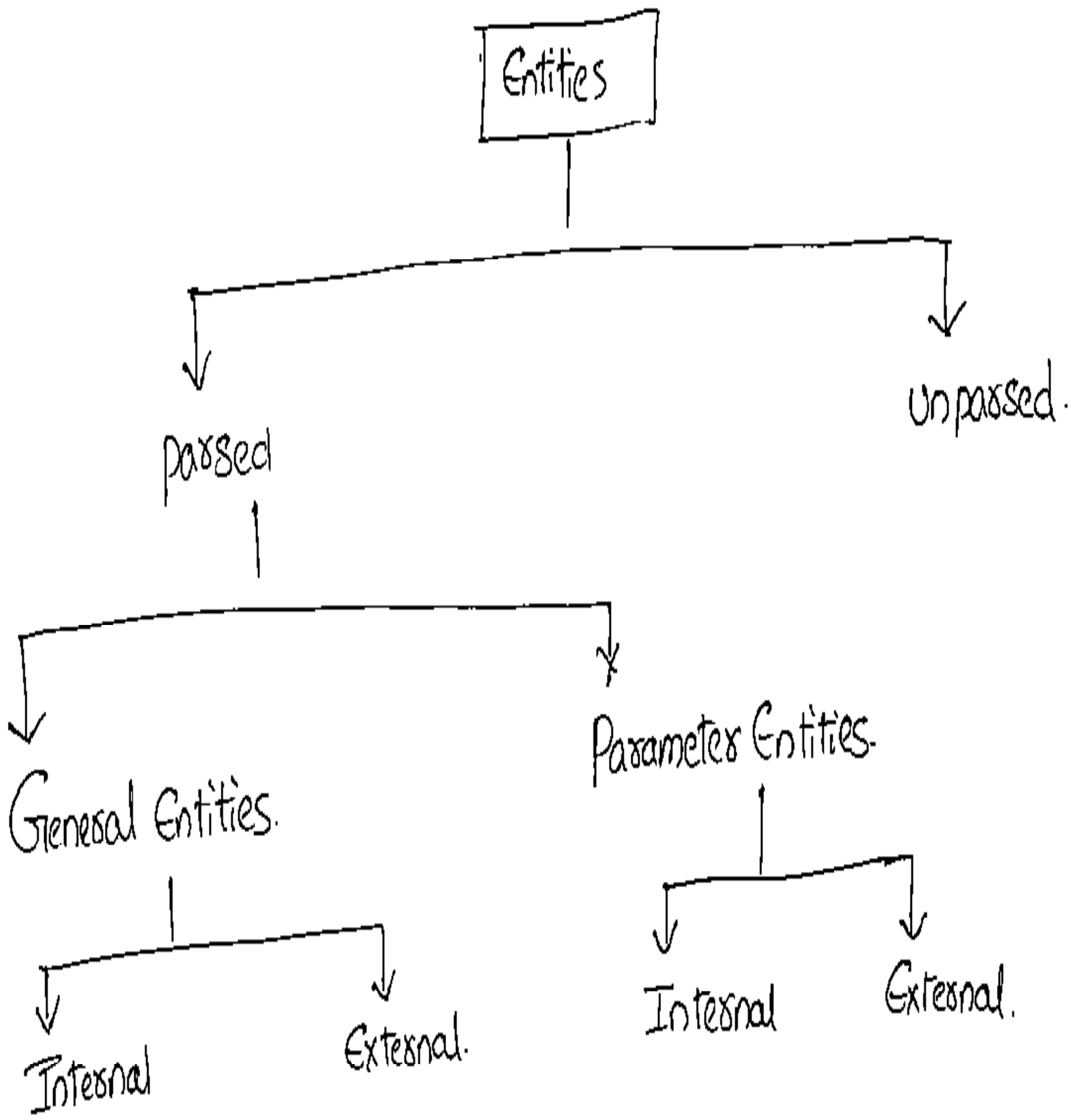
=) Entities are also used to describe the Unparsed Content.

Parsed =) Reusability purpose these are using.

I.) General Entities :- These type of Entities are declared in the dtd document and used in the xml document.

These Entities are Categorized into two types.

a) Internal Entities.
b) External Entities.


Entities
parsed
Unparsed.
General Entities.
Parameter Entities.
Internal
External.
Internal
External.

ⓐ Internal Entities :-

⇒ The Content referred by these Entities is described in the document declaring Entity.

⇒ To declare these type of Entities in the DTD Document

Syntax :-

<!ENTITY entityName="Content">

⇒ To use in the xml document

Syntax is.

&entityName;

For eg :-

<!ENTITY Copyrights

"javadays – 2012 - Nit">

⇒ In the 'xml document' &Copyrights; ✓

<Copyrights> &Copyrights; </Copyrights>

ⓑ External Entities :-

⇒ External Entities refers to the Content i.e, into an external file or document.

Syntax :-

<!Entity entityName SYSTEM "file location">

⇒ To use the External Entities in the xml document

Syntax :-

&entityName;

Point to Note:- We are generally accommended to use Internal Entities. But we go for External Entities in the following two Cases.

a) If the Content is very huge.
b) If the Content is frequently modified.

II. Parameter Entities :

=) These type of Entities are declared in the DTD and used in the DTD document itself.

=) These types of Entities are again Categorized into Two types.

a) Internal Entities
b) External Entities.

a) Internal Entities :

Syntax for declaring :-

```
<!ENITY % entityName "Content">
```

Syntax for using :

```
% entityName;
```

for eg :-

```
<!Entity (ENTITY) % decorativeTags "font|bli|ulp">

<!ELEMENT body
(#PCDATA| % decorativeTags; |form| p0e)">
```

b) External Entities:-

External Entities refers to the content. i.e into an external file or document

<! ENTITY % entityName SYSTEM "dtd location">

for using the syntax:-

% entityName;

Point to note:- we give only one dtd document to the xml document.

unpassed Entities :-

These type of entities are used to describe the content like 'image', 'gif', 'jpeg'

These type of entities are declared in DTD and accessed in the XML DOCUMENT using the attribute of type entity (or) Entities.

Syntax:-

<! ~~Ent~~ ENTITY entityName SYSTEM "fileContent" NDATA notationName" >

Notation :- Notation. Consists of addition text to better describe the context, like encoding's context types.

Syntax:-

<! NOTATION notationName SYSTEM "description">

for eg <! NOTATION gif SYSTEM "image/gif">

2) Now to declare an unparsed entity.

<!ENTITY banner SYSTEM "javabanner.gif" NDATA "gif">

<!ATTLIST Cinema mybanner ENTITY #REQUIRED>

<Cinema mybanner = "banner">

30/8/12

Preparing XML DOCUMENT STRUCTURE:-

In to the xml Document, we can have the following tags.

1) processing tag :- This tag is an optional tag in the xml document, if used it should be first tag.

The processing tag Commence with "?"

<?ML <attributes> ?>

The attributes Can the following

a) version :- This attribute is mandatory.

b) encoding :- This is an optional attribute, when we can Specifys the encoding's like UTF-8, UTF-16

c) Stand alone :- This attribute is again an Optional attribute, whose values Can be yes/no. meaning whether the xml Document depends on an external resource or not, In case of yes, the attribute value is yes, otherwise no

<? xml version = "1.0"
encoding = "UTF-8" Standalone = "no" ?>

2) <! DOCTYPE>

=) DOCTYPE standards for document type declaration.

=) This tag is used to associate the xml document with the DTD document

=) This tag can be used in 3 ways.

a) Internal Doctype:

Here the DTD code is into the xml document

Syntax:

<! DOCTYPE root elementName [DTD code]>

b) External Doctype:

Here the DTD code is into a seperate document, which refer from the xml document.

=) for External Doctype we have two syntax's.

ⓐ <! DOCTYPE rootelementName SYSTEM "dtd code location">

ⓑ <! DOCTYPE rootelementName SYSTEM (PUBLIC "FPI (formal public Identifier)" "DTD file url">

=) FPI is divided into 4 parts each part is seperated from the other using //

ⓐ - / +

ⓑ Name of the organization who has given the DTD.

ⓒ what for the DTd is given.

ⓓ two letters language code.

Note:- In the above Syntax the DTD file url need not be reachabl

for eg:- The following is the external DOCTYPE for web.xml document

<! DOCTYPE web-app PUBLIC "-//SUN MICRO SYSTEMS, INC
· //DTD web Application 2.3//EN".
" http://java.sun.com/dtd/web-app-2-3.dtd" >

(3). Internal and external DOCTYPE:-

This DOCTYPE is randomly used, the Syntax is

<! DOCTYPE rootelementName SYSTEM/PUBLIC [External DTD]
[Internal DTD] >

point To Note:- Here the internal DTD can override the external DTD.

=) Now we go for preparing Markup language Specification, where we are interested to persist.

=) Our requriement is to persist Cinema details like Cinema names, hero, heroine, budget. villan, Distributor, Copyrigh

=) The attributes used will be dtrid, prid, talk

=) Copy rights is an Entity.

=) for this requriment the root element is Cinemas, into which we will have

Cinema elements distributor.
elements Copyrights.

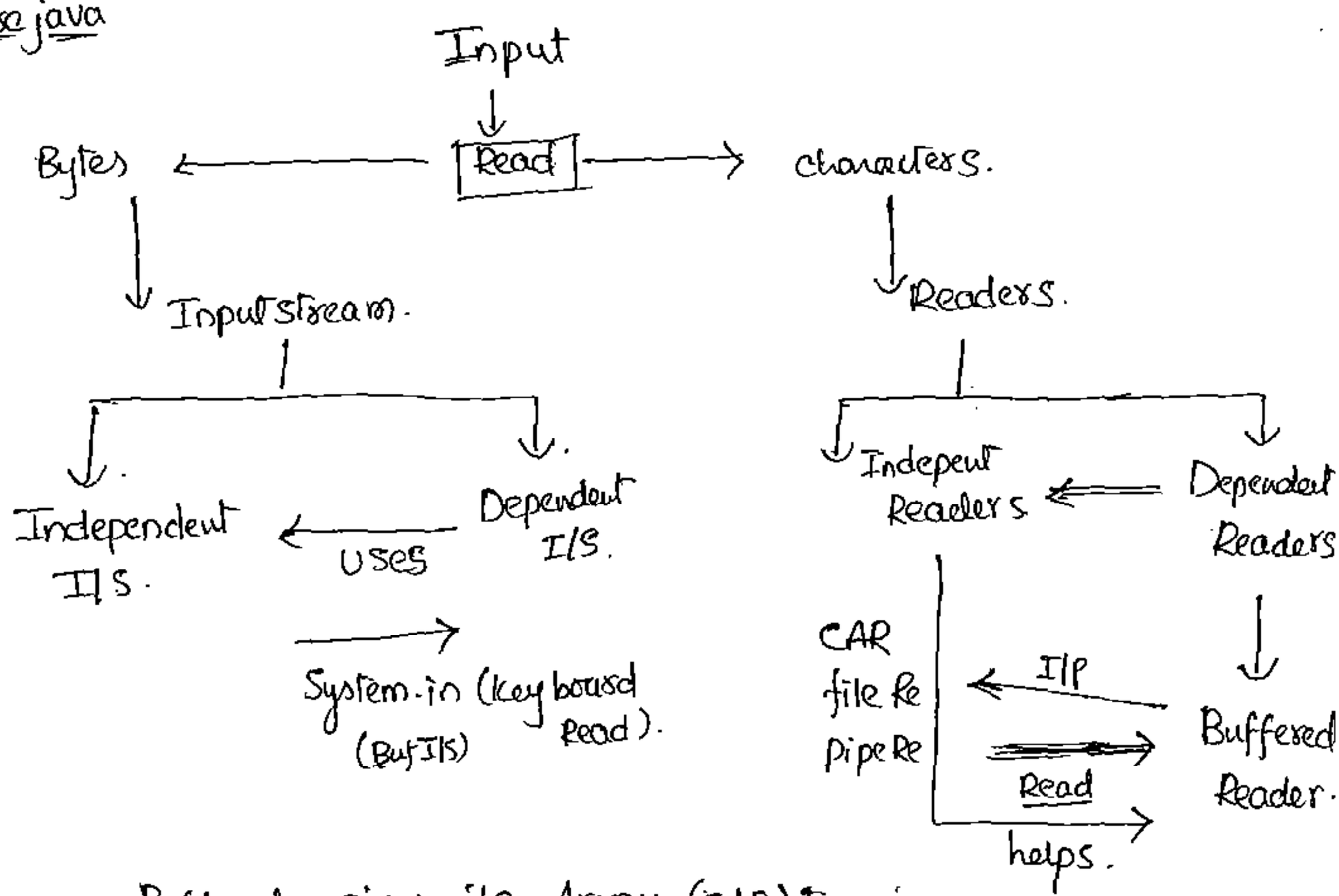
=) The child element for Cinema are name, hero, heroin, budget, villan, and their values are text.

=) The child elements for distributor is distributor name and distributor location, and the attribute is distid.

=) Preparing the DTD document for the above requirement.

```
<! ELEMENT cinemas (cinema *, distributory, Copyrights)>
<! ELEMENT Cinema (name, hero, heroin, budget, villan)>
<! ELEMENT distributor (distributor name, distributor location)>
<! ELEMENT Copyrights (# PCDATA)>
<! ELEMENT name (# PC DATA)>
<! ELEMENT hero <# PC DATA)>
<! ELEMENT heroin (#PC DATA)>
<! ELEMENT budget (#PC DATA)>
<! ELEMENT villan (# PC DATA)>
<! ELEMENT distributor name (# PC DATA)>
<! ELEMENT distributor location (#PC DATA)>
<! ATTLIST cinema disid ID (# PC REQUIRED)>
<! ATTLIST cinema proid IDREF (#IMPLIED)>
<! ATTLIST cinema talk (hit | superhit | flop) "hit">
<! ATTLIST distributor disid ID (# REQUIRED>
<! ENTITY Copyrights "Copyrights - nareshit - 2012">
```

Core java



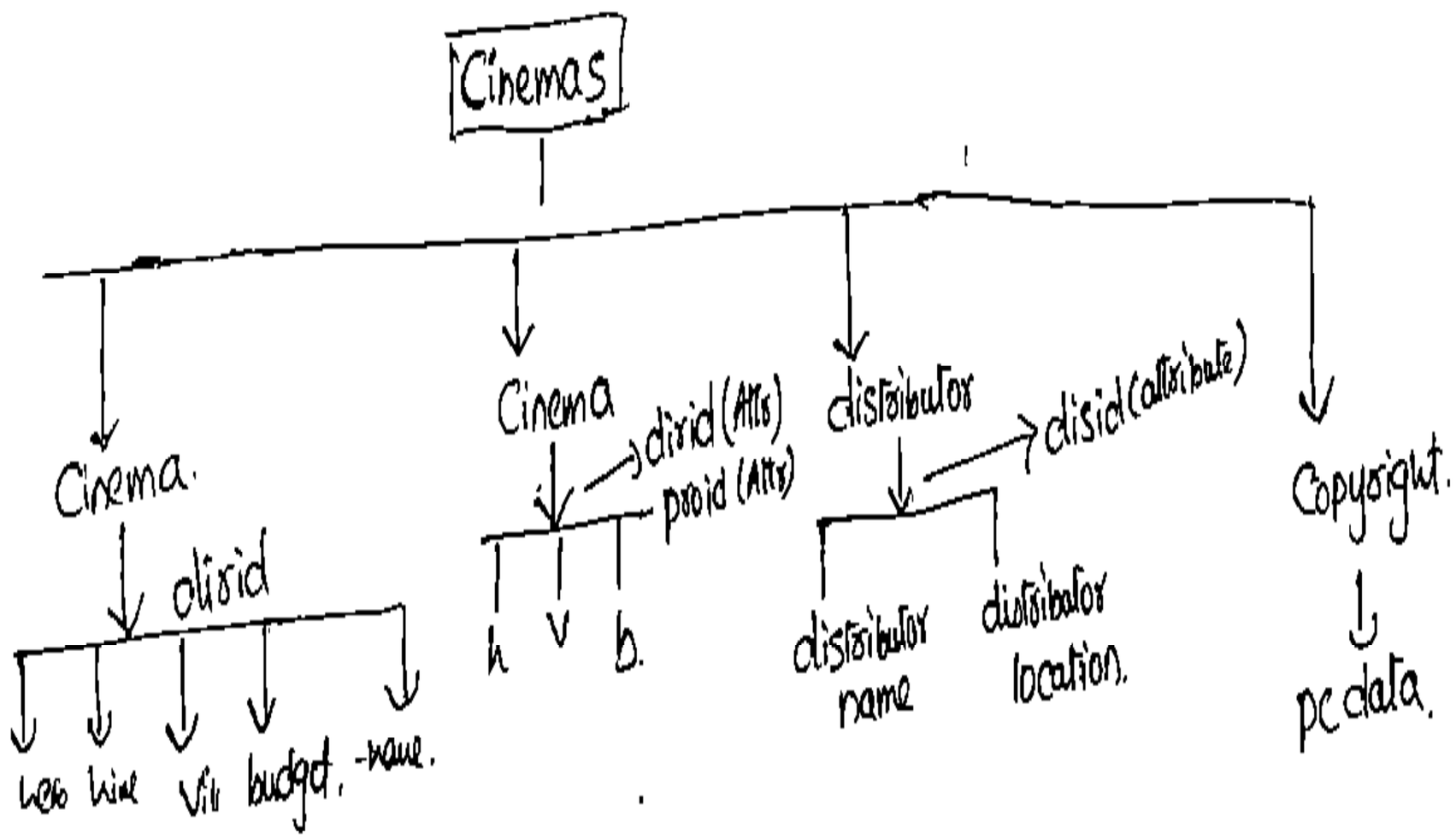
⇒ Buffreader gives i/p Array (CAR) To Indepent Reader.

⇒ when you use "System.in" internally it gives "Buffered I/p Stream"

⇒ B/S dependent, we have to give. Indepent i/p.

⇒ public static void main (String... s)
↓ args → variablename
instead of array.

pv Set DoCument locator(locator l)
{ this.l = l
}
↓ obj under whose Content behaviour location.


Cinemas
Cinema.
cinid
loc
lic
vil
budget.
name.
Cinema
cinid (Attr)
proid (Attr)
h
v
b.
distributor
disid (attribute)
distributor name
distributor location.
Copyright.
pc data.

31/8/12

Cinemas.xml

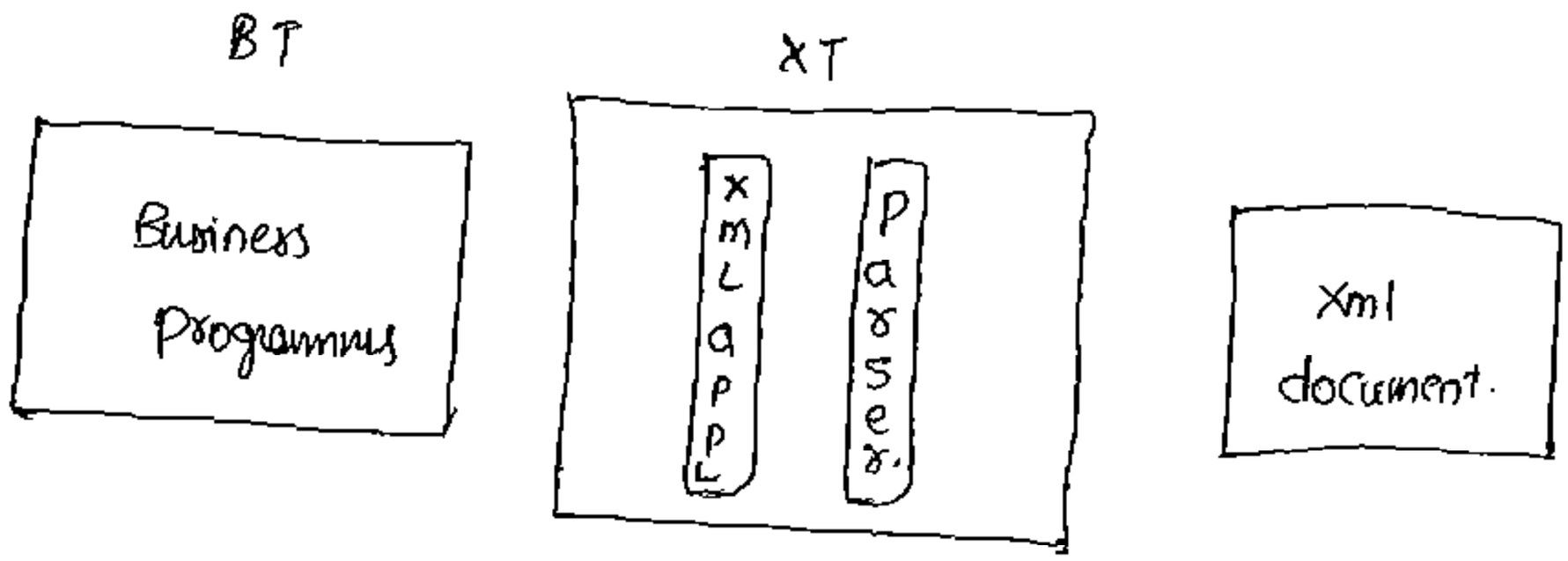
```
<?xml version="1.0"?>

<!DOCTYPE cinemas SYSTEM "Cinema.dtd">

<cinemas>
                    directorid          producerid
<cinema  dirid="d1"  proid="d2"
    talk="Superhit">
      <name> Sholay </name>
      <hero> Veeru </hero>
      <heroin> Basanthi </heroin>
      <budget> 20 </budget>
      <villan> Gabbar </villan>

  </cinema>

<cinema  dirid="d2"  talk="hit">
      <name> Shiva </name>
      <hero> Nag </hero>
      <heroin> Amala </heroin>
      <budget> 30 </budget>
      <villan> Raghuvaran </villan>
      </cinema>

<distributor  distid="dist1">
<distributorname> Dilraj </distributorname>
<distributor location> Nizam </distributor location>
  </distributor>

<Copyright> © Copyrights ; </Copyrights>

  </cinemas>
```

⇒ Save that document as Cinema.xml

⇒ In the above we have prepared XML ML Specification, and now we are interested to put that xml document into Action.

⇒ So that we can perform CRUD Operations like Creating, Reading, Updating, and deleting.

⇒ So for performing these Operations, we go for using parsers and Nowdays we have the parser Implementation into almost all the languages. (programming languages)



⇒ XML Applications :-

⇒ Xml application, is a System, implemented using a programming language, to perform Operations on the xml document. Operations like reading, creating. And also implementing 'business activities'.

⇒ parser provides a standard 'abstraction' and 'Infrastructures' to the Xml application to access the xml data, that is into Xml Document.

=) As (per) the Parser Implementations are used To access the xml documents, They are Categorized as follows.

=) On the Basis of validation

a) Nonvalidating Parsers

b) Validating Parsers

=) Non validating parsers are those parsers which check only the welformedness of the xml document

=) These parsers doesn't verify aganist the DTD OR XML Schema document.

=) Welformedness Means, it To Satisfy the following rules.

a) The xml document should have one 'root' element

b) The elements should be properly nested, meaning if there are child elements, without closing the child elements we shouldn't Close the parent element.

c) The attributes should be (') or (") quoted.

d) The Entities should be properly declared.

Validating Parsers :

=) Validating parsers check the welformedness of the xml document, as well as verify aganist the DTD or Schema document (xml).

=) we have DTD and XML schema validators.

=) On the basis of exposing the XML data to the application we have two approaches.

a) Object Tree Based approaches

b) Event Based Approaches

i) pull

ii) push.

DOM (Document Object Model) :-

=) DOM parser Architecture is given by w3c.

=) DOM parser uses object Tree based approach for exposing the XML Content.

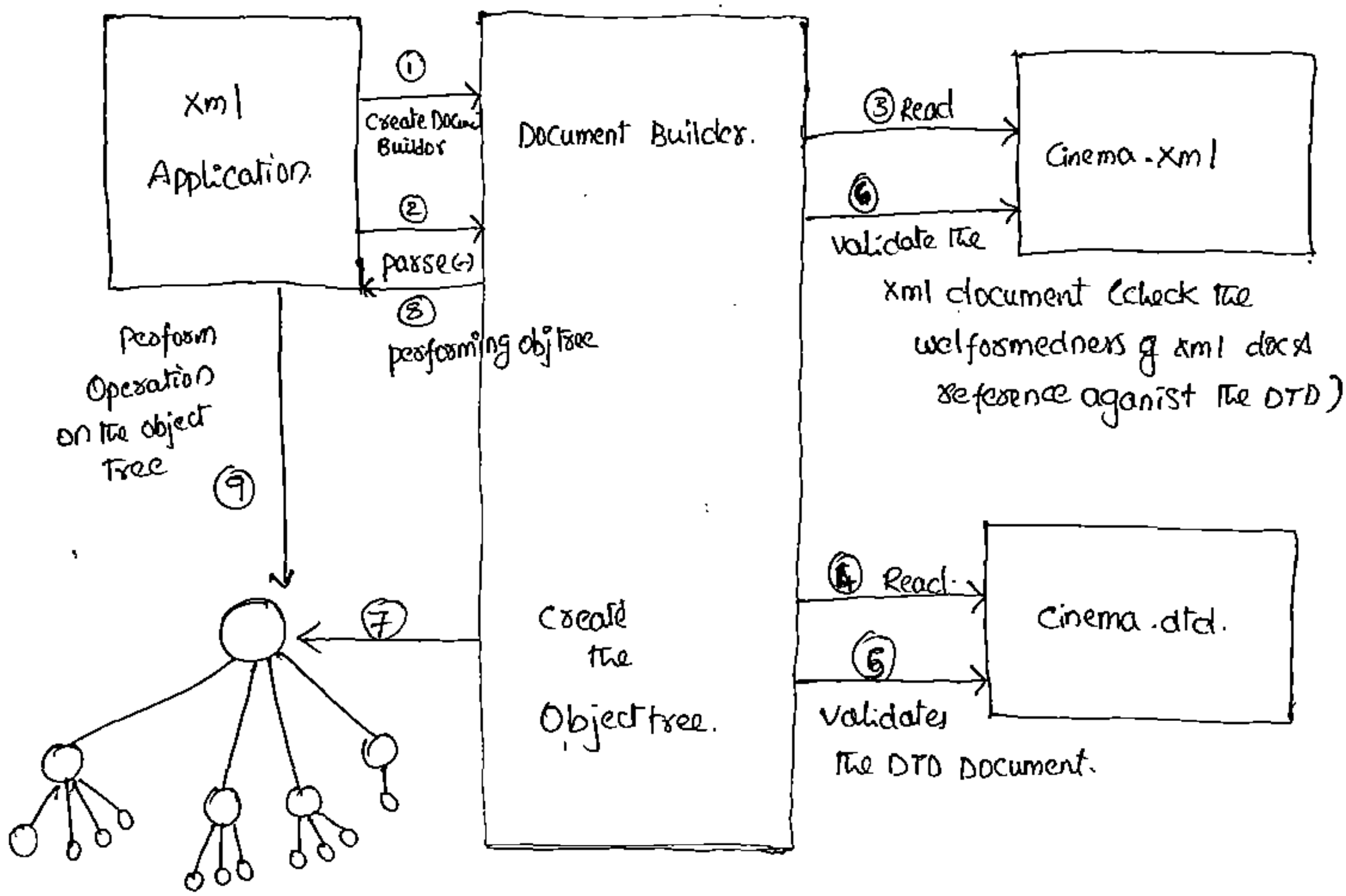
=) DOM parser Supports validating the DTD as well as XML Schema.

=) we have various levels of DOM parser, like DOM level 3 (Similar to the term versions)

=) DOM parser Supports performing all the four operations like create, read, update, and delete on the XML document.

1/sept/2012

DOM Parser Architecture



Note :-

They are various implementations of DOM parser in To java, like xml4j from IBM, Xerces from Apache, parserv2 from Oracle corp. Upto JDK 1.4 we were using JAXP, which was requiring additional Configuration, from Jdk 15, we have Xerces implementation builtin.

=) Requirement is To read data from the xml document, so we write an xml Application, where we will be using the libraries that area into javax.xml parser and org.w3c.dom packages.

ReadCinemaNames.java:-

```
pk com.nareshit.dom;

import javax.xml.parsers.*;
import org.w3c.dom.*;
import java.io.*;

public class ReadCinemaNames {

    psvm( ) throws Exception

    /* step1
    Get the DocumentBuilderFactory
```

=) .... This is used to obtain the parser, that produces DOM Object trees.

=) .... we have the following behaviours into the abstract class

new Instance(): Is a factory behaviour that returns DocumentBuilderFactory

newDocumentBuilder(): This behaviour returns the DocumentBuilder.

```
    */

DocumentBuilderFactory dbf = DocumentBuilderFactory.
                newInstance();
                dbf.setValidating(true);

    /* Step2: The obtain the DocumentBuilder
where DocumentBuilder is used to get the DOM DOCUMENT
    Object from the xml document */
```

```
DocumentBuilder db
    = dbf. newDocumentBuilder();

/* Next Step is to parse the XML DOCUMENT
So we use the parse (InputStream) behaviour, of DOCument
Builder, and this behaviour is overloaded */

FileInputStream fis = new FileInputStream ("cinema.xml

Document doc = db.parse (fis);
{/*
}// main
}// class.

Note:- D:\xml> SetClassPath = jdk 1.6.0
D:\xml\ java com.nareshit.dom.ReadCinemaNames. */ }

// Get the Document Element

/* use the behaviour getDocumentElement(), the Output
Type of this behaviour is org.w3c.dom.Element

=> where Element is an interface subtyping from
org.w3c.dom.Node
=> where Element describe abt an element
and Node is the Generic Type */
Element cinemas = doc.getDocumentElement();
```

/* Now use the behaviours of Node interface to navigate the document

=) we have behaviours like [2 method & those 2 navigate]

NodeList getChildNodes()

NodeList getElementsByTagName(String)

NodeList is an Interface describing the ordered Collection of Nodes

getChildNodes() behaviour returns a list of Nodes even the ignorable whiteSpaces.

getElementsByTagName(String) returns a list of Element class of Object, that equals to the given input name

*/

NodeList

Cinemas_childs = Cinemas.getChildNodes();

/* The NodeList I/f has the following behaviour

getLength(): returns the number of Nodes in this list

item(int): returns the index th element of the list

*/

S.O.pln(Cinemas_Childs.getLength());

for(int i=0; i< Cinemas_Childs.getLength(); i++)

{

Node Cinemas_child = Cinemas_childs.item

/* get the Node Type, and then verify whether it is an Element node or not, If it is Element Node, then get the Element Name.

=) for getting the NodeType, use the behaviour getNodeType() of Node, and use the Constant to verify whether is Element Node or not

Note:

Node.ELEMENT_NODE

*/

Cinemas_child.getNodeType() = Node.ELEMENT_NODE)

{

// now use getNodeName() out type is String

String element_name = cinema.getNodeName();

/* Now Check whether the ElementName is Cinema or NOT, if yes get the Cinema Element */

if (element_name.equals("Cinema")) {

Element cinema = (Element) Cinemas_child;

// get the childNodes

NodeList

Cinema_childs = Cinema.getElementByTagName("name");

```
Node Cinema-Child = Cinema-Childs.item(0);

Cinema-child.getChildNodes();

Node name-Child = name-Childs.item(0);
(or) Node name-child = cinema-child.getFirstchild();
/* Now use getNodeValue() behaviour to get the
Node value */

S.o.pln (name-child.getNodeValue());

} if
} if
} for.
} //main
} //class.
```

Steps to Compile and execute the application.

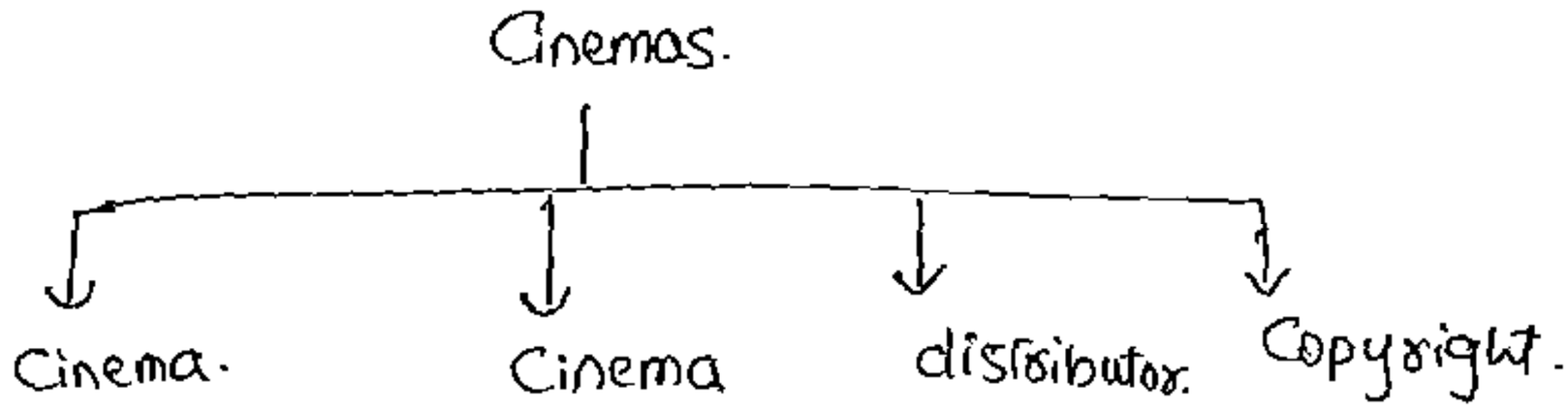
1) No need to have any additional classpath settings if we are using vendor specific libraries, say IBM, this requires additional classpath setting for 'xmL4j.jar'.

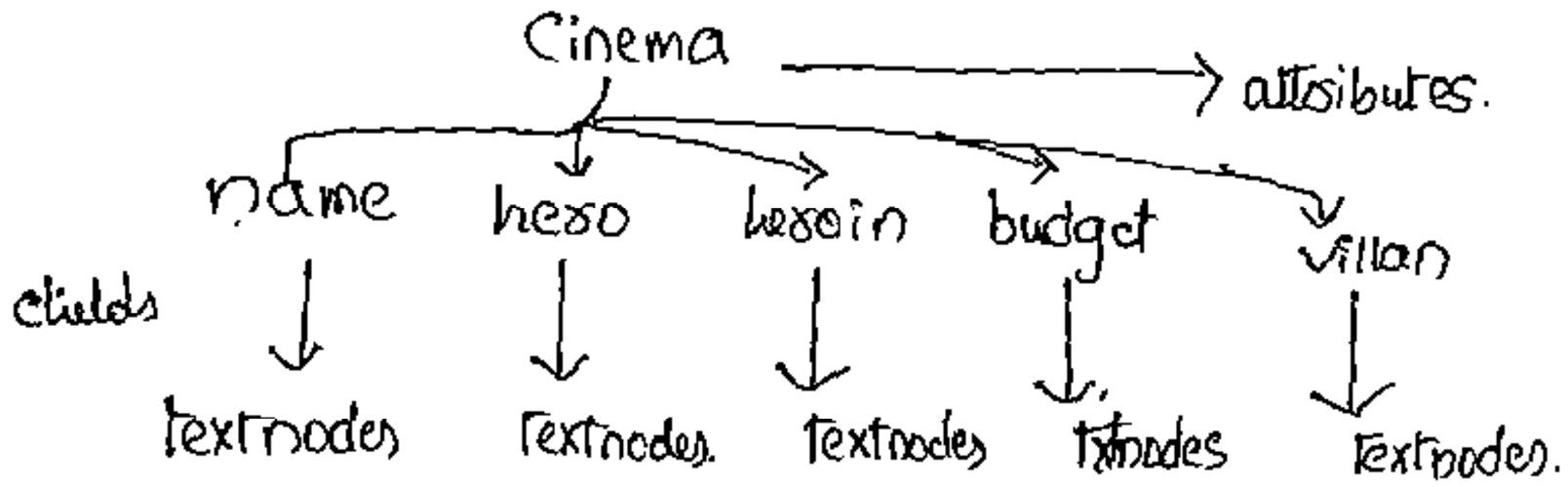
2) just set the path for

   set path = C:\jdk 1.6.0\bin; %path%

3) javac -d. ReadCinemaNames.java

4) java com.nit.com.ReadCinemaNames





Q ⟹ insert the 'Cinema' (above) in b/w Cinema and distributor.

InsertCinemaDetails.java

```
import static java.lang.System.*;   [out.println(" ");]
pk com.nit.com;                          ⇓
                                   no need write
import javax.xml.parsers.*;          S.o.pln.
import org.w3c.dom.*;
import java.io.*;
import javax.xml.transform.*;
```

/* we imported this pk to get the transformer object, from the TransformerFactory, as the transformer object is used to transform the source tree into result tree.
*/

```
import javax.xml.transform.dom.*;

/* this package we import to instantiate the DOMSource,
 this acts like a placeHolder, for the transformation.
 of DOM Tree */

import javax.xml.transform.Stream.*

/* this package we impost to create a StreamResult,
 where it acts like a placeholder for the transformation.
 of the result */

public class InsertCinemaDetails
{
  psvm( ) throw Exception
}
DocumentBuilderFactor dbf = DocumentBuilderFactory.newInstance
                                                    ();
    dbf.setValidating(true);
DocumentBuilder db = dbf.newDocumentBuilder();
    FileInputStream fis = new FileInputStream("Cinema.xml");
      Document doc = db.parse(fis);
/*  To load from the classpath
ClassLoader cl = InsertCinemaDetails.class.getClass
                Loader();
```

```
InputStream is = Cl.getResourceAsStream("Cinema.xml");
    Document doc = db.parse(is);

/* In this application we wanted to insert another Cinema
   element to the Cinemas tree, read the values of dirid
   name, hero, heroin, budget and villan from the Standard
   - Input */

    InputStream is = System.in;
    InputStreamReader isr = new InputStreamReader(is);
      BufferedReader br = new BufferedReader(isr);

/* BufferedReader br = new BufferedReader(new InputStream
   - Reader(System.in)); */

  out.Println("Enter dirid");
    String dirid = br.readLine();

  out.Println("Enter name");
  String name = br.readLine();

  out.Println("Enter hero");
  String hero = br.readLine();
  out.Println("Enter heroin");
  String heroin = br.readline();
```

```
Out.println ("Enter budget");
  String budget = br.readLine();
Out.println (" Enter villan");
  String villan = br.readLine();
  // Create Elements.

/* use the behaviour createElement (String) the output Type
   of this behaviour is Element and this behaviour
   is into Document intf */

Element cinema-ele = doc.createElement ("cinema");
Element name-ele = doc.create Element ("name");
Element hero-ele = doc.create Element ("hero");
Element heroin-ele = doc.create Element ("heroin");
Element budget-ele = doc.create Element ("budget");
Element villan-ele = doc.Create Element ("villan");

  /* Now Create Text Nodes using the behaviour
  Create Text Node (String) */

Node name-txt = doc.Create Text Node (name);
Node hero-txt = doc.Create Text Node (hero);
Node heroin-txt = doc.Create Text Node (heroin);
Node budget-txt = doc.Create Text Node (budget);
Node villan-txt = doc.Create Text Node (villan);
```

// Now append the textNodes to the respective
// elements using the appendChild (Node)

Name – ele. appendChild (name – txt);
(hero) Name – ele. appendChild (hero – txt);
(heroin) Name – ele. appendChild (heroin – txt);
(budget) Name – ele. appendChild (budget – txt);
villan – ele. appendChild (villan – txt);

// Now append the above elements to Cinema element.

Cinema – ele . appendChild (name – ele);
Cinema – ele . appendChild (hero – ele);
Cinema – ele . appendChild (heroin – ele);
Cinema – ele . appendChild (budget – ele);
Cinema – ele . appendChild (villan – ele);

Cinema – ele . setAttribute ("disid", disid);

Element Cinemas = doc . getDocumentElement ();

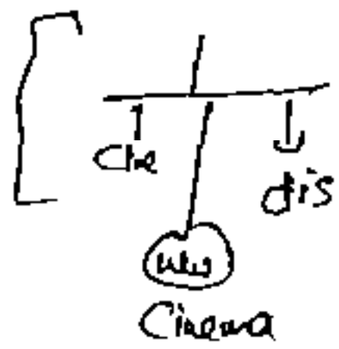
NodeList

Cinemas_Childs = cinemas . getElementByTagName ('distributor');
small

Node distributor = Cinemas_childs . item (0);

// Use the insertBefore (Node new, Node ref)

Cinemas . insertBefore (cinema – ele, distributor)



```
TransformerFactory tf = TransformerFactory.newInstance()
  Transformer t = tf.newTransformer();
    DOMSource ds = new DOMSource(doc);
    StreamResult sr = new StreamResult(System.out);
          t.transform(ds, sr);
      }
  }
```

Structure :-

Cinema.dtd
Cinema.xml
ReadCinemasDetails.java
InsertCinemasDetails.java

```
C:\xml> tree /f
package        D:-
                 |   Structure
                 |--> com
                      |--> nit
                           |--dom
                                ReadCinemaNames.class
                                InsertCinemaDetails.class
C:\xml> javac -d . InsertCinemaDetails
C:\xml> java com.nit.dom.InsertCinemaDetails.class
```

(*) Do the following modifications to the above application

// to write a file.

=) StreamResult sr = new StreamResult(new File("newCinemas.xml

=) for Indenting we can use Sunpropexieroxy api in this case

import com.sun.org.apache.xml.internal.serialize.*;

OutputFormat of = new OutputFormat(doc);

of.setIndenting(true);

XMLSerializer xs = new XMLSerializer(System.out, of)

xs.serializer(doc);

(above is replace of transformerfactory)

=) Apart from the above. we can also use XML4j parser given by IBM

if we use this parser do the following modifications to your application.

(replace the following code with)

DocumentBuilderFactory, DocumentBuilder and Document with the following code.

import com.ibm.xml.parser.*;

// Instantiate the parser

Parser p = new Parser("myerrors");

```
TxDocument doc = p.readStream(new FileInputStream
        ("cinema.xml"));
```

=> // Instead of getting TransformerFactory, Transformer, DOMSource, StreamResult

write the following code

```
doc.printWithFormat(new OutputStreamWriter(System.out));
```

Note: additionally set classpath for xml4j.jar.

Q => Requirement is to replace the child elements values.

Ans

```
package com.nit.dom;
import static java.lang.System.*;
import javax.xml.parsers;
import org.w3c.dom.*;
import java.io.*;
import com.ibm.xml.parser.*;
import com.sun.org.apache.xml.internal.serialize.*;
class ReplaceCinemaDetails {  psvm() throws Exception {
```

=> Requirement is to replace the cinema child elements values

=> do the following modifications to your previous application.

```
// the above is same as pa

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

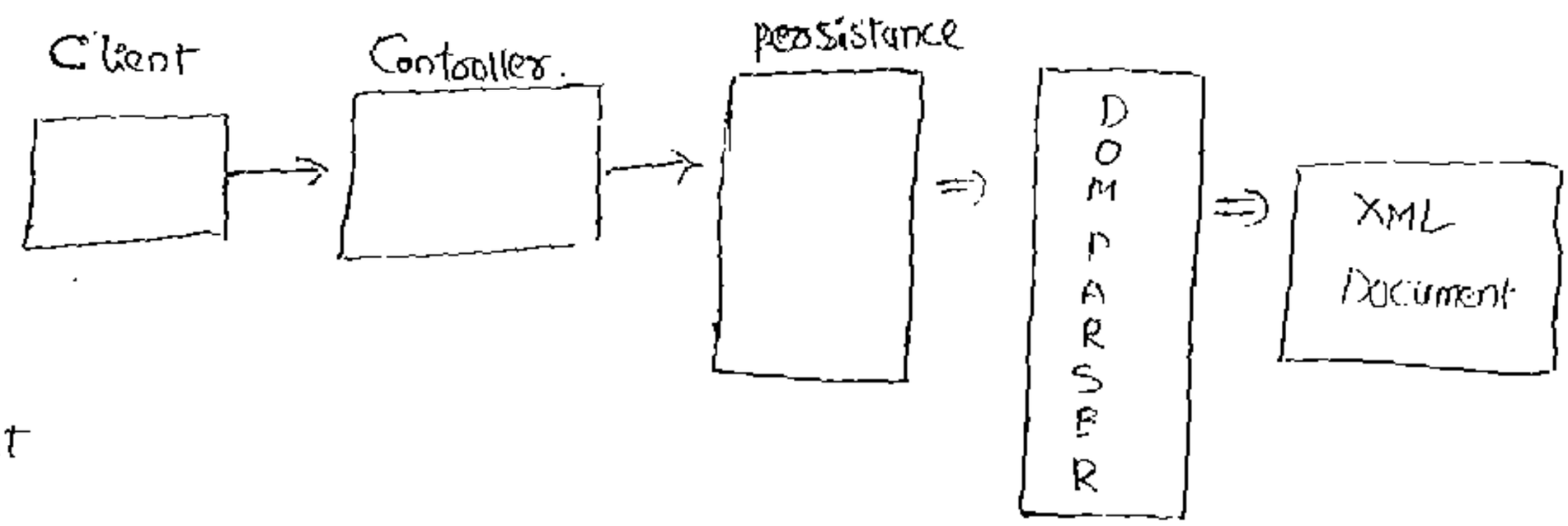
```
NodeList nl = cinemas.getElementsByTagName("cinema");
    for (int i=0; i<nl.getLength(); i++)
    {
        Element existingcinema = (Element) nl.item(i);
if (existingcinema.getAttribute("diaid").equals(s[1]))
    {
        // now use replaceChild(-,-)
Cinemas.replaceChild(cinema_ele, existingcinema);
        // below code is also same as program application.
```

Q) Requirement of replace only the heroin.

=) Do the following additional code of the previous above logic (first application)

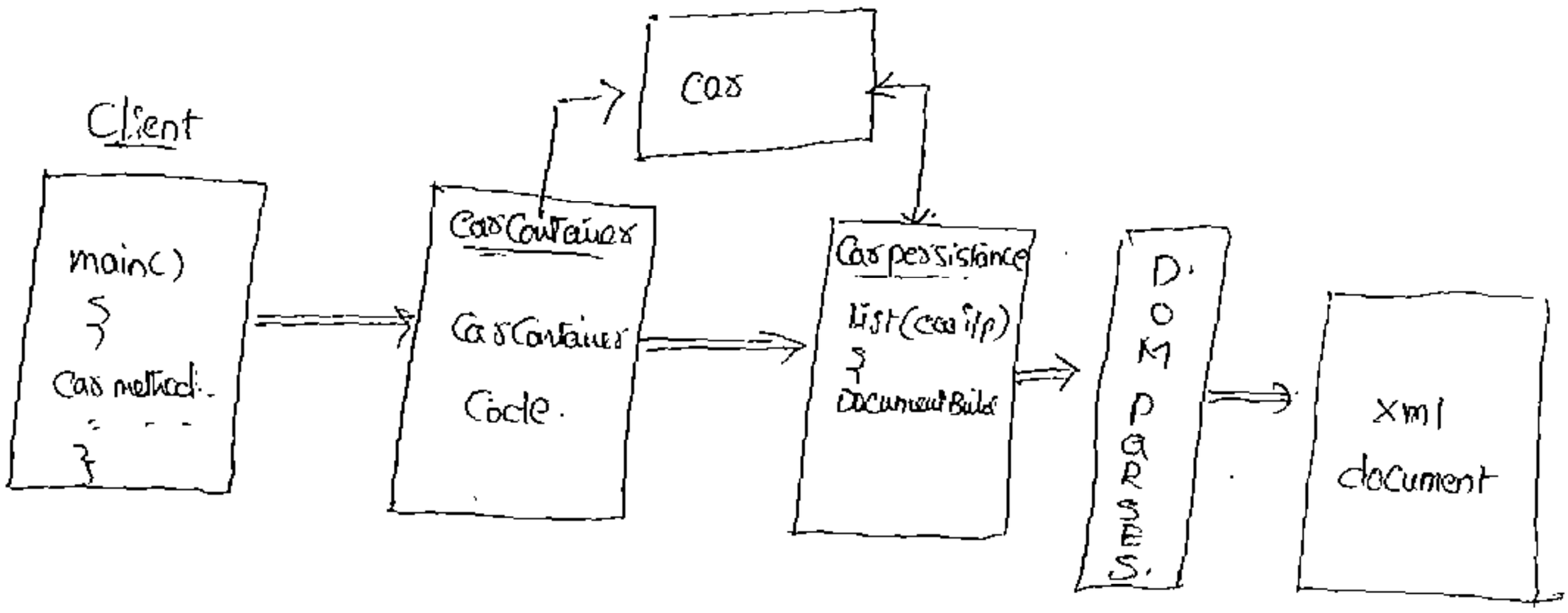
```
// after the if stmt
Node heroin = cinema.getElementsByTagName("heroin").item(0);
Node heroin_txt = heroin.getChildNodes().item(0);
```

5/09/12



Requirement

```
<cars>
<car>
<manfacture name> Audi </manfacture name>
<model name> A7 </model name>
<price> 100000 </price>    </car>  </cars>
```



Client :-

```
Package com. nit. Escate ;

public class car {

    String manufacturerName, modelName;
    double price;

Public void setManufactureName(String manufactureName)
    {
    This. manufactureName = manufacturer Name ;
    }

public String getManufacturerName()
    {
    return this. manufacturer Name.
    }

public void setModelName(String ModelName)
    {
    This. modelName = ModelName.
    }

public String getModelName ()
    {
    return modelName ;
    }
```

```
public void setPrice(double price)
{
    this.price = price;
}

public double getPrice()
{
    return price;
}
```

CarController.java.

```
pk com.nit.create;
import java.util.*;
import java.io.*;
import static java.lang.System.*;

public class CarController
{
public List<Car> readCars() {
List<Car> cars = new ArrayList<Car>();
BufferedReader br = new BufferedReader(new InputStream
-Reader(System.in));
do {
Out.println("Enter manufacturerName");
String manufacturer = br.readLine();
Out.Println("Enter ModelName");
String modelName = br.readLine();
```

```
Out.Println ("Enter Price");
String price = br.readline();

Car c = new Car();

c.setManufactureName(manufacturerName);
c.setModelName(modelName);
c.setPrice(Double.parseDouble(price));

Cars.add(c);
Out.Println ("Are u interested for adding
        another Car ?");
} while (br.readLine().equals("yes"));

// return the List
return Cars;

} // readcar.

public void createCar() {
List<Car> cars = readCars();
CarPersister cp = new CarPersister();
cp.saveCars(cars);
} // createCar
}
```

CarPersister :-

```
pk com.nit.create;

import static java.lang.System.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import java.io.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
import com.sun.org.apache.xml.internal.serialize.*;

public class CarPersister {
public void saveCars(List<car> cars) throws Exception
  {
    DocumentBuildersFactor dbf = DocumentBuildersFactor.
                                 newInstance();
    DocumentBuilder db = dbf.newDocumentBuilder();
    Document doc = db.newDocument();
Element cars_ele = doc.createElement("cars");
    doc.appendChild(cars_ele);
    for(Car c : cars) {
Element car_ele = doc.createElement("car");
Element manufacturersname_ele = doc.createElement("manufacturer
                name");
Element Model_ele = doc.createElement("modelname");
```

```
Element price_ele = doc.CreateElement("price");

Node manufactureName_txt = doc.CreateTextNode(c.getManufactureName());
Node modelName_txt = doc.CreateTextNode(c.getModelName());
Node price_txt = doc.CreateTextNode(c.getPrice()+" ");

manufactureName_ele.appendChild(manufactureName_txt);
modelName_ele.appendChild(modelName_txt);
price_ele.appendChild(price_txt);

car_ele.appendChild(manufactureName_ele);
car_ele.appendChild(modelName_ele);
car_ele.appendChild(price_ele);
cars_ele.appendChild(car_ele);
}

OutputFormat of = new OutputFormat(doc)
    of.setIndenting(true);
XMLSerializer XS = new XMLSerializer(new FileOutputStream
    ("cars.xml"), of);
    XS.serialize(doc);
}
}
```

6/9/12

C:\xml> javac :. java com.nit.create.ClientTestCase Compile.

Q) Do the following modification in ReplaceHeroin.java

after

RemoveCinema.java

```
if ( cinema.getAttribute("disid").equals(s[0]))
{
    cinema.removeChild(cinema);
/* remaining stuff to print to the output Console is Same */
}
}.
```

Note:- So far we have Completed Understanding the DOM parser where we have performed operations like

Creating, Reading, updating, Deleting

SAX (Simple API for XML) Parser.

=) SAX parser follows event based approach to expose the xml data to the xml application.

=) This parser is used to perform only Read Operation.

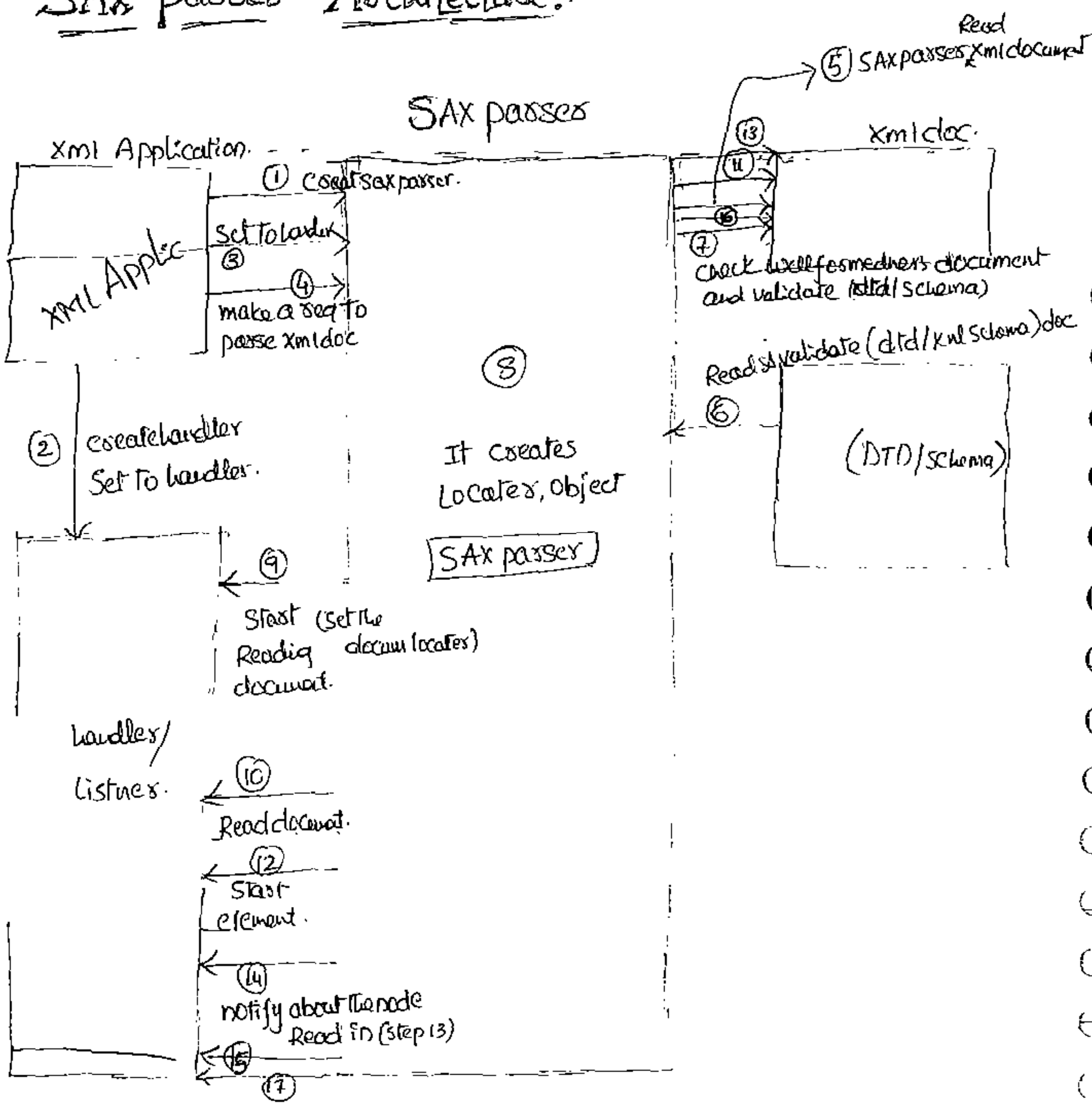
=) Using this parser we can perform Simple search Operation

=) It uses PUSH model

=) Unlike the DOM parser, this parser makes the xml data available to the xml application as and when it reads the xml document.

Point to note :- with respect DOM parser we have issues with Memory Mgt, as the DOM Tree should loaded into the Active Memory. we have also performance issues as we have many Associations, method Calls, and objects Created, while using DOM parser.

SAX parser Architecture:



⑪ Read root element

⑬ Read next node.

⑮ Read ends Root element.

⑥ Read End element.

⑰ End document.

1) Creating the SAX parser.

2) Create the Required Handler.

3) Set the Handler to the parser.

4) Request the parser to parse the xml document

5) Sax parser reads the xml document.

6) Sax parser reads the validates the DTD / Schema document.

7) Sax parser checks the welformed ness of xml document and validates it aganist the DTD / schema

Point to Note (SAX 2.0 parsers are Schema as well as DTD validators)

8) Create the locator object (ValueObject, to get the Line Numbers or the position of the character in the doc)

9) Sets the Locator to the Handler

10) Starts reading Document

11) Reads the Root element

12) Starts reading the Element.

13) Reads the next Node

14) Notifies about the node read in Step 13

15) Reads End of Element

16) Reads End of Root Element

17) Reads End of Document.

7/9/12

=) Now let us use SAXParser and Read the xml.

Note to point :- JAXP provides Standard abstraction for ~~xml~~ accessing the xml Content, in our java Application. where JAXP Supports parser. Like DOM, SAX, STAX (Streaming API for xml)

=) To Read the Data from the xml Document

we have the following steps

1) Get the SAXparserFactory.

```
SAXParserFactory factory = SaxParserFactor - newInstance();
```

2) Get the SAXParser using the above factory

```
SAXParser parser = factory.newSAXParser()
```

3) Create a listner i.e Handler Implementation.

4) Request to parse the xml document

```
parse.parse(InputStream, Handler)
```

Note:- we Can use vendor Specific Implementation, say Xerces Implementation from apache.

```
Parser parser = ParserFactory.makeParser("org.apache.xerces.
    parsers.SAXParser");
```

// creating a listner, either by implementing the DocumentHandler interface or by extending an Adapter org.xml.sax.helpers.HandlerBase

// Setting the Listner to the DocumentHandler

parser.setDocumentHandler(<ListenerImp>

// Request to parser

parser.parse(<XmlDocument>)

// Have set the classpath for xercesimpl.jar.

Q) // SAXParser Sample.

```
package com.nit.Saxparserread;
import org.xml.sax.*;
import javax.xml.parsers.*;

public class SAXParserReadSample
{
    psvm(String...s) throw Exception.
{
    SAXParserFactor factory = SAXParserFactor.newInstance();
        SAXParser parser = factory.newSAXParser();
        MyListener ml = new MyListener();
Parser.parser(new.java.io.FileInputStream("cinema.xml").ml);
    }//main
}// class
```

small

=) Now implement listner Application

=) Earlier we were implementing an interface named DocumentHandler, now we use the Adapter class DefaultHandler, where we can use the following behaviours like

Void startDocument() :- This behaviour is used to receive the Notification of the start of Document.

Void endDocument():- This behaviour is used to receive the Notification of the end of the Document.

Void startElement (String nameSpaceUri, String localName, String qualifieldName, Attributes at)

Void endElement (String nameSpaceUri, String localName, String qualifieldName)

=) The above two behaviours are used to receive the notifications of passing the start element and end element.

Void characters(char[] c, int index, int length):-

This behaviour is used to get the Notification of the Character data in the element.

Void ignorableWhiteSpace (char[] c, int index, int length):-

To receive notification of white spaces.

void processingInstruction(String s, String s1) :-

To receive notifications of the processing instruction's.

setDocumentLocator (Locator l) :- To get the Locator Object, which is value Object, and it is used to get the line message, characters, positions in the xml document.

// My Listener :-

```
pk com.nit.Saxparserread:
import org.xml.sax.*;
import javax.xml.parsers.*;
import org.xml.sax.helpers.*;

public class MyListner extends DefaultHandler
{
    public void setDocumentLocator(Locator l)
    {
        this.l = l;
    }
    Locator l;

    public void StartDocument()
    {
        S.o.pln("Document Parsing Started");
    }
    public void endDocument()
    {
        S.o.pln("Document parsing Ended")
    }
    public void StartElement(String uri, String lname,
        String qname, Attribute at)
    {
        S.o.pln("Element = '+qname+' : : parsing Started');
        for (int i=0; i<at.getLength(); i++)
        {
            S.o.pln("Name : : "+ at.getLocalName(i));
            S.o.pln("value : : " + at.getValue(i));
        }
    }
```

```
public void endElement (String uri, String lname, String
                                                qname)
    {
      S.o.pln ("Element::"+qname+"::parsing ended");
    }

public void Character (char[] c, int index, int length)
    {
      S.o.pln ("Text::"+newString (c, index, length));
    }
} // class.
```

STAX PARSER

(Streaming Api for Xml)

=) This parser uses Event Based approach to expose the XML Content to the XML Application.

=) There are two approaches to access the XML Data, using the Stax parser

a) Cursor based Approach
b) Event based Approach

=) Unlike SAX, this parser uses pull model.

=) when Compared dom and SAX, Stax is Faster

=) we can use this parser for reading Partial data from the Xml document.

⇒). D: cd xml

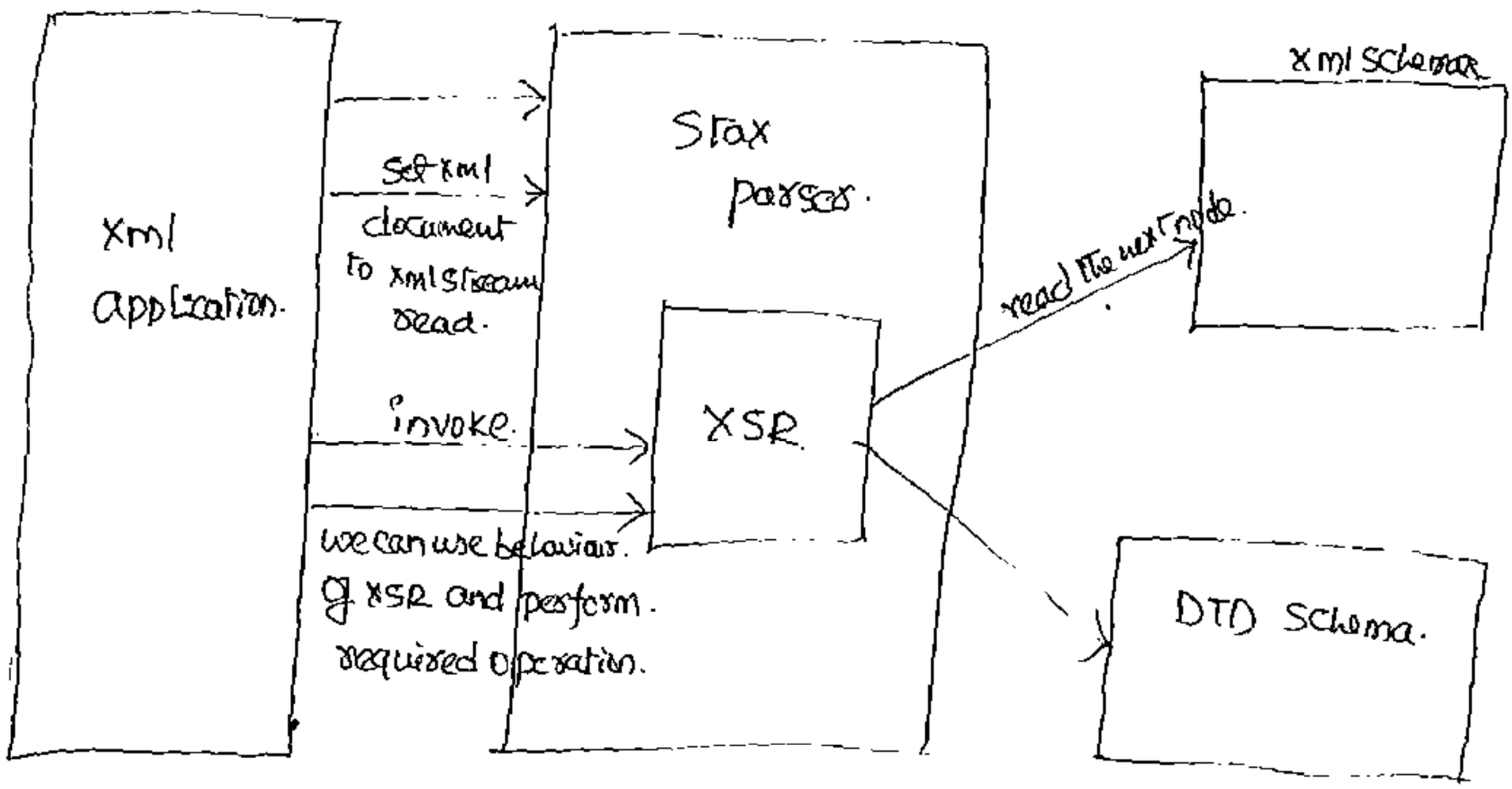
D: xml: javac -d - *.java.

Structure :- SaxParserReadSample.java.
MyListner.java.
Cinema.xml
Cinema.dtd.

⇒) we can find the use of this parser mostly in mobile editions.

Note:- Using this parser we can perform Operations like Read and write, this parser doesnot Support Modifying and Deleting the Nodes.

⇒) Now let us implement a Simple example where we read the XML Document



flow - diagram

Steps are

a) Get the XMLInputFactory

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

b) Get the XMLStreamReader, using above factory

```
XMLStreamReader reader
= factory.createXMLStreamReader(new FileReader
("cinemas.xml"));
```

c) Use the behaviours like next() or nextTag(), of XMLStreamReader, where these behaviours outputType if int, describing the event parsed.

d) The use the Constants given in XMLStreamConstants like

START_DOCUMENT
END_DOCUMENT
START_ELEMENT
END_ELEMENT
CHARACTERS

and then use the behaviours of XMLStreamReader and read the xml data like

getLocalName()
getAttributeName(int)
getAttributeValue(int)
getText()
getAttributeCount()

StaxRead :-

```
package com.nit.stax;
import javax.xml.stream.*;
import java.io.*;

public class StaxRead {
  psvm(String...s) throws Exception
  {

XMLInputFactory factory = XMLInputFactory.newInstance();
XMLStreamReader reader = factor.CreateXMLStreamReader(new
        FileReader("cinemas.xml"));
// move the cursor to the next node
        int event = reader.next();
  while(event !=
  XMLStreamConstants.END_DOCUMENT) {
  if(event = XMLStreamConstants.START_ELEMENT)
  {
      // get the ElementName
      // get the Attributes, name & values.
      S.O.pln("Element :: "+reader.getLocalName()+
                " :: started");
      for(int i=0; i<reader.getAttributeCount(); i++)
      {
        S.O.pln("Name :: "+ reader.getAttributeValue(i));
      }
  }

  else if (event == XMLStreamConstants.END_ELEMENT)
  {
```

```
S.o.pln ("Element :: " + reader.getLocalName() + " :: Ended");
        }
  Else if (event = XMLStreamConstant.CHARACTERS)
        {
         So.pln ("Element :: " + reader.getText());
        }
        event = reader.next();
      } // while
    } // main
  } // class.
```

=) To Compile & Execution the application, if we have installed with jdk6, we no need to do any Classpath Setting, just Set the path for c:\jdk1.6.0\bin

Otherwise Set classpath for "Staxapi.jar"

(*) <u>Write Operation Using Stax</u> :-

<u>Steps</u> :-

a) Get the XMLOutputFactory

```
XMLOutputFactory factory = XMLOutputFactory.newInstance()
```

b) Get the XMLStreamWriter

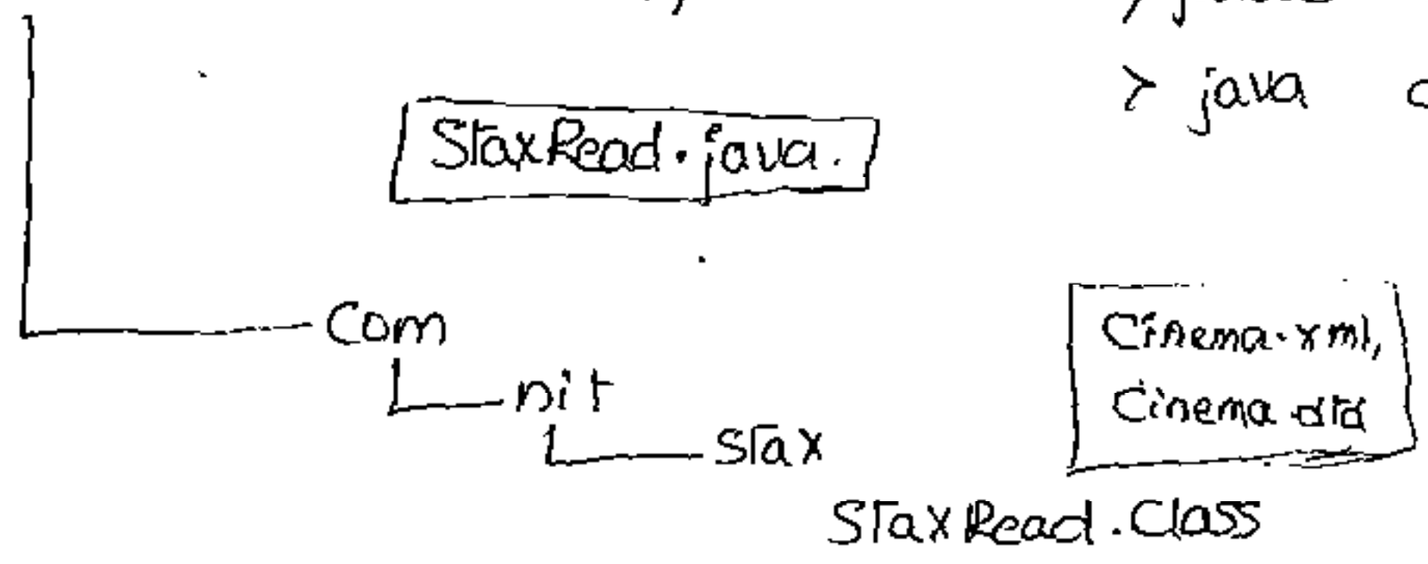
```
XMLStreamWriter writer = factory.createXMLStreamWriter(new
        FileWriter("movies.xml"));
```

D:\xml\stax > tree /f

> javac -d . StaxWrite.java

> java com.nit.stax.StaxRead.ja

Compile :



D:\xml\stax java com.nit.stax.StaxRead.java.

---

c) The use the behaviours of

XMLStreamWriter like

WriteStartDocument()
WriteEndDocument()
WriteStartElement(String localname)
WriteEndElement()
Write ~~Text~~ Characters(String)
WriteAttribute(String name, String value)

StaxWrite :-

```
package com.nit.stax;
import javax.xml.stream.*;
import java.io.*;
public class StaxWrite
{
    psvm(String... s) throws Exception
}
XMLOutputFactory factory = XMLOutputFactory.newInstance();
XMLStreamWriter writer = factory.createXMLStreamWriter
    (new FileWriter("movies.xml"));
        writer.writeStartDocument();
```

```
writer.WriteStartElement("movies");
writer.WriteStartElement("movie");
writer.WriteAttribute("m1", "value1");
writer.WriteStartElement("name");
writer.WriteCharacter("Life is beautiful");
writer.WriteEndElement();
writer.WriteEndElement();
writer.WriteEndElement();
writer.WriteEndElement();
writer.WriteEndDocument();
writer.flush();
    } //main
} //class.
```

=) In the previous two application we have used STAX parser for performing Reading and writing Operations.

=) To Access the XML Data, the Approach we Followed was Cursor Based Approach. So here we have read the XML Document from top to Bottom.

=) Now we follow the Second approach for accessing the XML Data ie Event Based approach.

=) Here also we move the Cursor to iterate over the xml document. In this Approach the XMLEventReader's nextEvent behaviour returns an XMLEvent Object, which describes the event data.

⇒ Steps to program for performing Read Operation.

a) Get the XMLInputFactory.

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

b) Get the XMLEventReader.

```
XMLEventReader Reader = factory.createXMLEventReader
(new FileInputStream("Cinema.xml"));
```

c) Get the XMLEvent Object

```
XMLEvent event = reader.nextEvent();
XMLEvent event = (XMLEvent) reader.next();
```

d) Then get the EventType using the behaviour of XMLEvent i.e getEventType(), and then Compare with the Constants of XMLEvent, where XMLEvent Subtypes of XMLStream Constants, So we have the following Constants.

START_DOCUMENT
END_DOCUMENT
START_ELEMENT
END_ELEMENT
CHARACTERS . . .

e) The get the StartElement Type, Similarly we can get EndElement type, Characters, and perform the required Operations.

... Imagine the requirement is to read the attribute values of dirid.

```
package Com.nit.stax.eventbased;
import javax.xml.stream.*;
import javax.xml.stream.events.*;
import java.io.*;
import java.util.*;

public class XMLEventReaderReader {
        psvm ( ) throws Exception
        {
XMLInputFactory factory = XMLInputFactory.newInstance();
XMLEventReader reader = factory.CreateXMLEventReader(new
    FileInputStream("Cinema.xml"));

    XMLEvent event = reader.nextEvent();
        while (reader. hasNext())
        {
            // get the Event type.
if (event.getEventType() == XMLEvent.START_ELEMENT)
        {
            // get the StartElement

StartElement StartElement = (StartElement) event;

    /* Use the behaviour getName(), This returns the
qualified name of the Element, so use getLocalPart()
behaviour, this behaviour return the ElementName, i.e
qualified Name minus NameSpace */
```

```
if (startElement.getName().getLocalPart().equals("cinema"
    {
        // NOW get the Attributes
        /* use getAttributes() behaviour this returns
            java.util.Iterator */

    Iterator it = startElement.getAttributes();
        while (it.hasNext())
        {
    Attribute attribute = (Attribute) it.next();
     if (attribute.getName()).getLocalPart().equals("dirid"))
      {
        S.o.pln(attribute.getValue());
      }
    if (attribute.getValue().equals(s[0]))
      {
        S.o.pln("cinema found");
      return;
      } // if
      } // while
      }
} } } }
```

⇒ <u>writing</u> Operation <u>using</u> <u>Stax's</u> <u>Event</u> <u>Based</u> approach :-

<u>Steps</u>:

a) Get the XMLOutputFactory

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

b) Get the XMLEventWriter :-

```
XMLEventWriter writer = factory.CreateXMLEventWriter
        (new FileOutputStream ("Courses.xml"));
```

Point To Note:- The add behaviour of XMLEventWriter takes input as an XMLEvent

c) Get the XMLEventFactory

```
XMLEventFactory event = XMLEventFactory.newInstance();
```

use the behaviour of XMLEventFactory, for Creating xmldocument, elements, attributes. ..

```
CreateStartDocument()
CreateEndDocument()
CreateStartElement (String prefix, String namespace,
                    String name)
CreateEndElement (String prefix, String namespace,
                  String name)
CreateAttribute (String name, String value)
CreateCharacters (String).
```

11/09/12

XSU [xml SQL Utility].

=) It is an utility API given by oracle app.

=) It provides libraries to read data from oracle db and write it to an xml Document

=) It also provides libraries to write the xml Content into the database

=) write an application to read from the db and write to an xml document.

```
package Com.nit.xsu.read;
  import java.io.*;
  import oracle.xml.sql.query.*;
  import java.sql.*;

public class ReadDel {
  public static void main (String ... s) Throw Exception
  {
  class.forName("oracle.jdbc.driver.OracleDriver");
Connection Con = //Instance DriverManager.getConnection("jdbc:oracle:thin:@
      localhost:1521:ord"; "user", "pwd")

  // Instantiate Oracle xml Query.
  OracleXmlQuery query = new OracleXmlQuery(con, "Select *
                                  from emp);
  // for transforming relation data into xml doc.
  // we use getXmlString() behaviour.
    String result = query.getXmlString();
  PrintStream ps = new PrintStream (new FileInputStream
```

```
("emp.xml"));

ps.println(result);
  System.out.pln ("Data retrieved and stored into xml document");

}
}
```

Note => To compile and execute the application set classpath.

for the following jar file

ojdbc 14.jar (oracle)

oracle xml sql.jar (naresh sir CD)

=> Store the xml content or data into database.

=> To to this we will be using Dom parser.

pk.com.nit.xsu.write:

```
import org.w3c.dom.*;
import com.ibm.xml.parser.*;
import java.sql.*;
import java.io.*;
import oracle.xml.sql.xml.*;

public class write Db
{
  psvm (String ... s) Throws Exception.
{
  Class.forName ("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection ("jdbc:oracle:
  thin @ localhost:1521:oracle", "uses", "pwd");

// Instantiate oracle xml save
```

```
OracleXMLSave save = new OracleXMLSave (con, "emp");

// Instantiate the parser

    parser p = new parser ("my error");
    Document d = p.readStream (new FileInputStream
                                    ("emp.xml"));

// use the behaviour insertXml(Document).

    int count = save.insertXml(d);
    S.o.pln ("Rows affected are:" + count);
  }
}
```

=) To Compile and execute this application additionally set the classpath for

xml14.jar
xmlparser.jar

=) Write a DTD Document.

```
<!ELEMENT RowSet (Row*)>
<!ELEMENT Row(Empno, Ename, Sal, Deptno)>
<!ELEMENT EmpNo (#PCDATA)>
<!ELEMENT EName (#PCDATA)>
<!ELEMENT SAL (#PCDATA)>
<!ELEMENT Deptno (#PCDATA)>
<!ATTLIST Row Num (DATA #Implied>
```

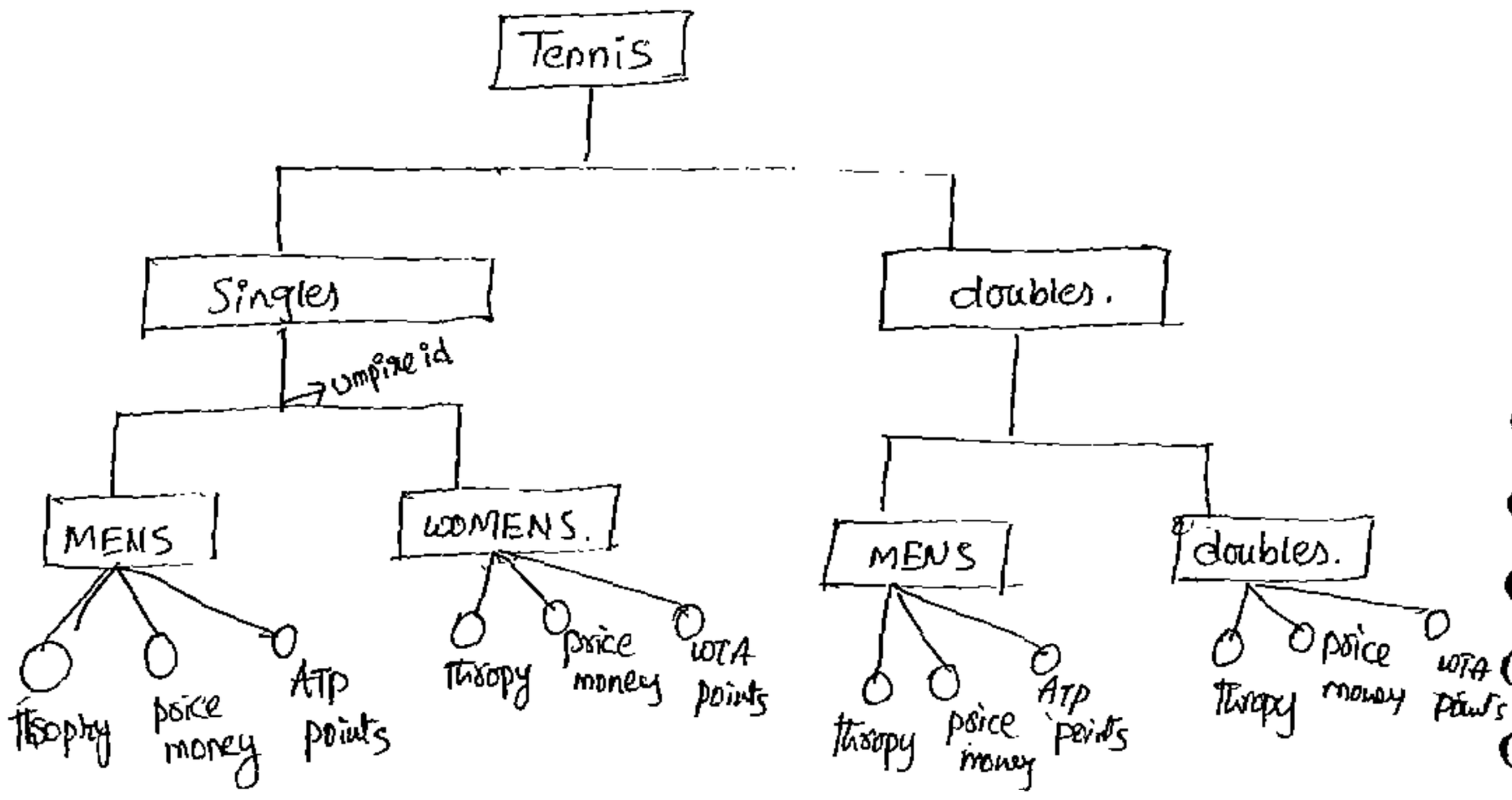
=) Save the file emp.dtd

=) In emp.xml

```
<!DOCTYPE RowSET System "emp.dtd">
```

15/09/12 WORK

Eg :-



Q) with in the time of 25 min prepare an xml dtd Specification and perform either read and or write operation using dom, Sax or Stax.

Ans

```
<!ELEMENT Tennis (Singles, doubles)
<!ELEMENT Singles (MENS, WOMENS)>
<!ELEMENT doubles (MENS, WOMENS)>.
<!ELEMENT MENS (#pc DATA)>
<!ELEMENT WOMENS (#pc DATA)>
<!ELEMENT MENS (#pc DATA)>
<!ELEMENT WOMENS (#pc DATA)>
<!ELEMENT Trophy (#pc REQURIED)>
<!ELEMENT priceMoney (#pc REQURIED)>
<!ELEMENT WTAPoints (#pc REQURIED)>
```

12/09/12

XSL [xml Stylesheet language]

=) XSL is a xml markup language

=) It is not a part of xml Specification, rather it is derived from xml Specification

Q) why XSL?

Ans =) To transform (Convert) one xml document (Format) to another xml document Format.

Note:- This type of requirement we encounter an Enterprise Application.

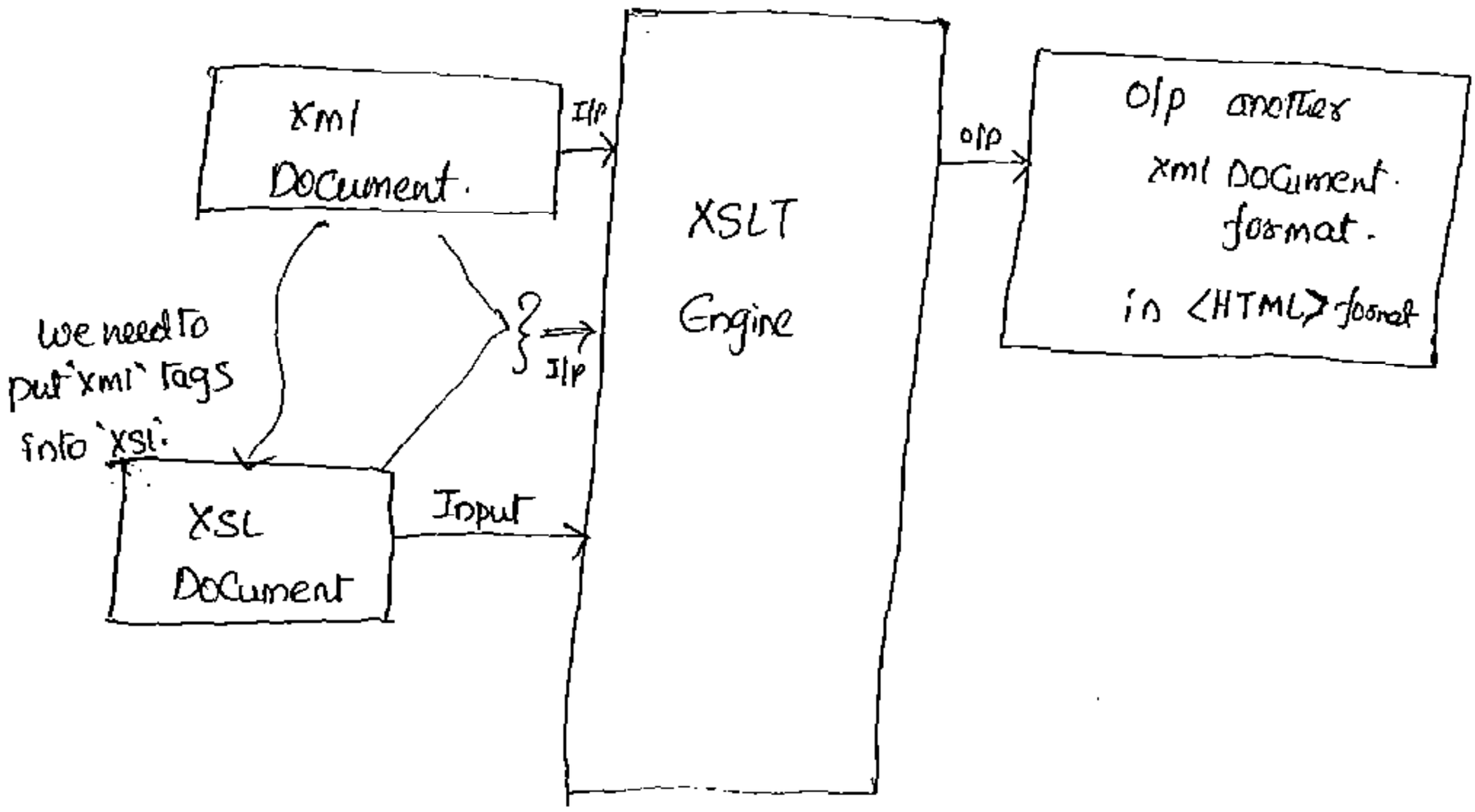
=) So here we have requirement like we have Some data into xml document, and we wanted to present in html format. To meet this type of requirement, we can uses parses, But using parsers and implementing the above requirement results to the following issues

a) lack of experience of the programmers affects the Application performance.

b) Application Development time increases, this results to increases in the Application cost, and finally results to less productivity.

c) It may be also very difficult to embed the code into Applications like web Browsers.

⇒ Now we require an Infrastructure, who takes the responsibility, to Convert one xml document format to another xml Document format, and this what is done by XSL.



⇒ Now we have to understand what we have to read from the Input XML Document, and what we have to enclose with in tags of XSL

⇒ So now we Commence learning the various XSL Tags

No DOCTYPE = NODTD is required here

⇒ <XSL : StyleSheet> :-

→ This element is the root element of an xsl document.

→ This element is enclosed with all the xsl tags, and import Statements.

For eg :- <xsl : StyleSheet version = "1.0"
xmlns : xsl = 'http://www.w3.org/1999/XSL/Transform'>
↓
name, spce [all xml tags in to "xsl" name space]

b) <XSL : Template>

⇒ This element is used to group Some instructions, to be executed by the XSLT engine.

⇒ This element is identified by the match pattern associated to it.

⇒ The match pattern is Specified using the match attribute of this element.

⇒ If we have <xsl : template match="/">
describes the document root.

⇒ In an XSL document we can have any number of XSL : Template elements.

c) <XSL : apply - templates>

⇒ This element gives instructions to Select all the nodes from the Input xml Document, and execute. match

⇒ This is done in 2 steps

1) Selects all the nodes from the input xml document using the Select pattern, which we describe using as select attribute.

2) Then identify the appropriate Templates or the Selected nodes and execute.

d) <XSL : value-of>

⇒ This element retrieves the values from the input xml document and writes it to the output.

⇒ So here we use Select pattern, described as

⇒ In case of attributes we prefix @

for eg:-

```
<xsl:value-of select="@dirid"/>      → it prints 'director' id
<xsl:value-of select="@name"/>       → it prints 'name' id.
```

Cinema.xsl

```
<xsl:stylesheet version="1.0"
<xmlns:xsl="http://www.w3.org/1999/XSLTransform">

  <xsl:template match="/">
  <xsl:apply-templates select="cinemas"/>
  </xsl:template>

  <xsl:template match="cinemas">
  <html><body><table border="1">
   <tr><th>DirectorId</th><th>Name</th><th>Hero</th><th>
   Heroin</th><th>Budget</th><th>Villan</th></tr>

  <xsl:apply-templates select="cinema"/>
      </table></body></html>
    </xsl:template>

  <xsl:template match="cinema">
      <tr><td>
          <xsl:value-of select="@dirid"/></td><td>
          <xsl:value-of select="name"/></td><td>
```

=) Sample to display

directorid, name, hero, heroin, budget and villan values.

=) write an XSL document using the above values.

~~<select>~~

< xsl : value-of select = "hero" /> < /td> < td>

< xsl : value-of Select = "heroin" /> < /td> < td>

< xsl : value-of Select = "budget" /> < /td> < td>

< xsl : value-of Select = "villan" /> < /td> < td>

< /xsl : template>

< /xsl : stylesheet>

=) save it as Cinema.xsl

=) In cinema.xml remove the following elements processing element, DOCTYPE and COPYrights

In the first line

< ? xml - stylesheet href = "cinema.xsl" type = "text/xsl" ?>

=) Then browse the Cinema.xml (open with Internet explorer)

=) Otherwise use the command

java.org.apache.xalan.xslt.Process - IN Cinemal.xml

- XSL Cinema.xsl - OUT Cinema.html

Set classpath for Xclass.jar.

e) <xsl: for-each> :-

=) This element iterates set of instructions Specified in the body part of the element, for ~~the~~ each of the Selected nodes.

=) The select pattern is Specified Using Select attribute.

=) So here this element performs two Operations

1) Selects the nodes from the input xml document.
2) executes the Specified instructions for each of the Selected nodes.

for eg:-

```
<xsl: for-each   Select = "cinema">
        <tr> <td>
<xsl: value-of   Select = "@ disid"/> </td> </tr>
<xsl: for-each>
```

Note:- Do the above modifications to our previous Cinema1.xsl document. And Save the Document as Cinema1.xsl, and Specify this path in the xml document, as value for the nref attribute. And then browse the xml document.

Note:- In to the <xsl: for-each> element, to specify the Sorting order we can even use

```
<xsl: Sort  name= " "  type= " "  Order= " " >
```

```
<xsl: import>
```

— xsl lets us to inherit another xsl document

2) To do the above we use xsl: import element,
This element takes an attribute named href. This attribute takes the path of xsl document

for eg:-

```
<xsl:stylesheet version="" xmlns:xsl="">

<xsl:template match="/">
<xsl:apply-templates select="Cinemas"/>
</xsl:template>
</xsl:stylesheet>
```

--- Save it as Cinema.xsl

=) Now in Cinemas1.xsl

```
<xsl:stylesheet version="" xmlns:xml="">
<xsl:import href="Cinemas.xsl"/>
<xsl:template match="">
<xsl:apply-templates select=""/>
</xsl:template>
</xsl:stylesheet>
```

=) Apart from the above tags or elements, xsl also supports Conditional tags, to be executed by the XSLT engine.
These tags takes a Condition, if the Condition resolves to boolean true, body part instructions are executed otherwise not.

<xsl:if>

=) It is an xml Condition tag, this element takes an attribute i.e test, whose value describes the Condition if it resolves to boolean true, the body of this element is evaluated.

for eg :-

```
<xsl:if    test = "Condition">
        // body Content
</xsl:if>
```

Note:- To build the Condition, we have to use xpath expressions.

13/09/12

for eg :- Do the following modification to the previous xsl document, here we have built an xpath expression.

```
<xsl:if    test = "self::Cinema [attribute::dirid = 'd1']">
        <tr> <td>

<xsl:value-of    Select = "@dirid"/> </td> </tr>
</xsl:if>
```

=) Save the document as Cinema2.xml

=) In the xml document href = "cinema2.xsl", and then browse the xml document.

Note:- Conditions are Specified in [ ] brackets

Note:- XSL doesn't Support else part.

f) <xsl : choose> :- This is Similar to Switch case in j

=) This element takes (one or more) xsl : when tags and optionally an xsl : otherwise tag.

Syntax :-

```
<xsl : choose>
<xsl : when test= "condition" />
<xsl : when test= "Condition 1" />
<xsl : otherwise = " " />
</xsl : choose>
```

Note :- xsl : when element takes an attribute named as test, to describe the Condition.

XPath

=) Xpath is an xml Specification.

=) Xpath is a Simple Language with few Constructs, to build expressions,

=) And these expression we refere as X-path expression, the xpath expression are resolved by X-path engine

=) Dom level 3 passers and XSLT engines are intelligent to resolve xpath expressions.

=) Xpath provides us the following Constructs (for performing operation) for Navigating via the nodes or node

Child ---->

This Construct describes all the childs of current nodes.

for eg:- child :: Cinemas

Self :--->

This Construct describes the current node

for eg:- Self :: Cinema

Parent :---> This Construct describes parent node of the current node.

for eg:- self :: parent

Ancestor :---> this Construct describes parent's, parent, parent --- nodes. ...

descendant ----> This Construct describes childes, childs child---, nodes.

attribute ---> This Construct describes all the attributes Of the current nodes.

following ...> Describes all the nodes next to this given node.

following-Sibling ...> describes the node next to the given node having a common ~~whose~~ parent

Preceeding ...> describes all the nodes before to the given node.

namespace : describes namespace.

ancestor-or-self :-

decendar-or-self :-

Apart from the above Constructs, Xpath Supports node

set functions like

Count (nodeset) :- returns the no. of nodes in the given nodeset

position() :- returns the position of the node in the nodes

Apart from Nodeset functions we also have boolean functions like.

not (boolean)
true()
False()

⇒ To get the nodeset properties we can use the following functions.

name()
local-name()
namespace-uri()

⇒ X path also Supports the following String functions

⇒ Sub String (String str, StartIndex, int length)

⇒ SubString-before (String str, String str1)

for eg :

Substring-before ("abc", "b") ⇒ a

⟹ Substring-after (String str, String str1)

for eg :

substring-after ("abc", "b") ⇒ c

⇒ Starts-with (String str, String str1) :-

returns true, if the first String starts with the Second String.

Contains (String str, String str1) :- returns true if the first String Contains the second string

String-length (String) :- returns the number of character of the given input String

normalize-space (String) :- removes the leading and trailing white spaces.

=) we also have number functions like

Sum ()
round()
Ceiling()
floor ()

2) Xpath also Supports Operators like Arithmetic, Relational

=
!=
>
>=
<
<=

+, -, *, mod(), div()

XML Markup Specification.

- XML Schema
- DTD
  - → Elements
  - → Attributes
  - → Entities
  - } passed
  - → notify → unparsed.

⇒ DOM (CRUD)

⇒ SAX (only Read.)

⇒ SMA
- CB
  - RW
- EV.
  - RW

⇒ XSU

⇒ XSL (xpath)

14/09/12

XML Schema

⇒ XML Schema is a new style of preparing XML markup Language Specification.

⇒ XML Schema uses XML Formatting Rules, unlike DTD which uses its own format in preparing the xml ml Specification

⇒ Differences between DTD and XML Schema, problems in DTD.

a) In DTD we Cannot Specify minoccurance and maxOccurance of an element.

b) DTD Document doesnot Support primitive data types,
In the DTD document we use #PC ~~data~~ DATA, CDATA, NMTOKEN, NMTOKENS.

c) DTD doesnot Supports Simple types, like Specify restrictions.

d) DTD doesnot Supports NameSpaces. (package concep in java)

e) As DTD uses its own formating Approach
it is not descriptive (not understanable)

f) DTD Supports Complex Types
Say we have an element named Emp whose child element is name, and the child elements of name, are fname and lname. So here name is child only element
In the DTD document we describe as

<!ELEMENT name (fname, lname)>

Now imagine we have another Element named dept, whose child element is also name, but it is TextOnly

The above types Cannot be described using DTD.

⇒ we go for XML Schema as it is extensible
⇒ XML Schema Supports Namespace } package
⇒ XML Schema Supports Data Types
⇒ XML Schema Supports to specify the Occurance of the elements.
⇒ Now we have to Understand, defining elements using XML Schema in an xml document.

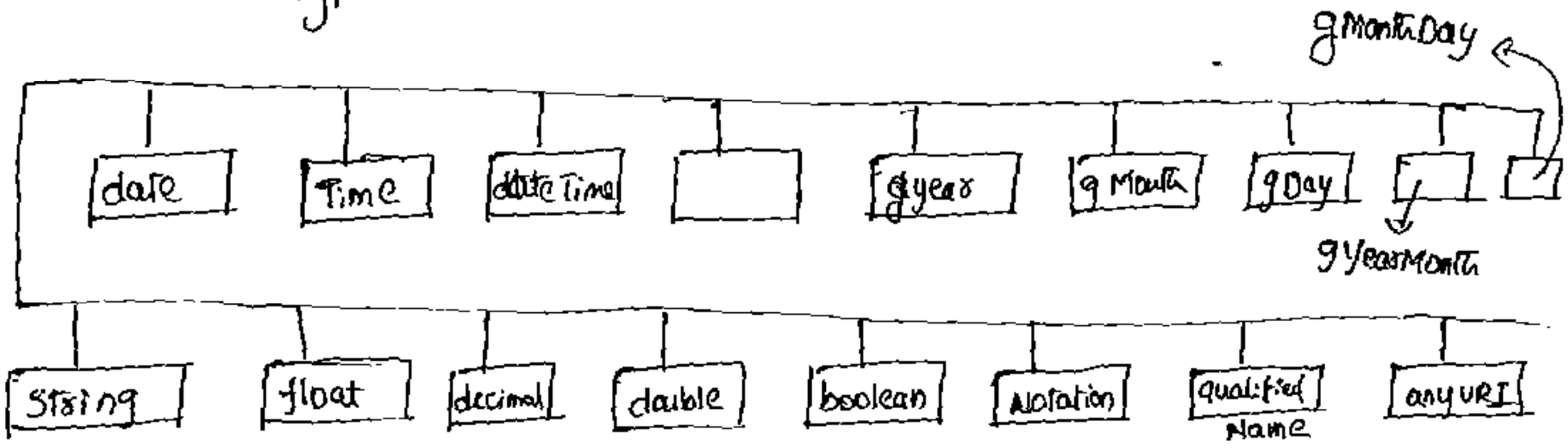
⇒ defining attributes
⇒ defining child elements
⇒ defining empty element

(recall what all we did with DTD.)

Note:- we Categorize the schema Elements into the following typ

a) Built-in-types
   - primitive types
   - derived types
b) user Defined types
c) Complex Types
d) Simple Types

⇒ Now First let us first list down the primitive and derived Types.



They are 19 primitive Supported by xml Schema and 25 derived Types.

1) String
2) decimal
3) duration
4) boolean
5) double
6) float
7) Qunamc
8) any URI
9) Notation
10) g year (gregorean)
11) g Month
12) g Day
13) g year Month
14) g Month Day
15) base 64 Binary
16) hex Binary
17) date
18) time
19) date Time

=) <u>The</u> <u>Derived Types are</u> :-

1) Normalized String
2) Token
3) Language
4) Name
5) NmToken
6) NC Name
7) NM Tokens
8) ID
9) IDREF
10) IDREFS
11) ENTITY
12) ENTITIES
13) Integer
14) Nonpositive Integer
15) Negative Integer

16) Non Negative Integer.
17) positive Integer
18) Long
19) Int
20) Short
21) Byte
22) UnSigned-Long
23) UnSigned Int
24) UnSigned Short
25) UnSigned Byte.

(XML Schema is built of XML and is Successor to DTD)

17/9/12

=) Now we Commence Creating an XML Schema Document

=) we use an element named as

```
< xsd : Schema
    xmlns : xsd = "http:// www.w3.org/2001/XMLSchema">
        . . . . . .
</xsd : schema>
```

=) In the above, we have used a NameSpace,

i.e http:// www.w3.org/2001/XMLSchema, Namespace is Similar to package in java application

Q) What is the purpose of Name Space?

Ans It is used to ~~[illegible]~~ identify ELEMENTS, ATTRIBUTES, Uniquely, in an XML document.

Q ⇒ why is this NameSpace so important?

Ans we can find its Importance, when we prepare an xml, document using a ML language or Multiple ML'S where these ML'S are processed using various ML processors, and the o/p of one ML processor is the i/p of another ML processor.

Note:- www.w3schools.com.

⇒ Understanding Namespace, includes two things.

a) Defining NameSpace for our ML

⇒ To define a Namespace for our ML, we use an attribute TargetNameSpace, into the Schema Element.

⇒ This attribute's value we define in an URL format and the URL may not be reachable.

for eg:-

<xsd:schema targetNamespace = "http://www.nit.com/MyNameSpace">

Unique identify attributes, Elements.

b) importing the NameSpace.

⇒ We import NameSpace to use elements and attributes in our Xml document.

⇒ We can import the NameSpace in two styles.

a) using prefix (General NameSpace)

b) without using prefix (Default NameSpace)

xmlns : <prefix> = "NameSpaceURL"

xmlns = "NameSpaceURL"

=) If we don't use prefix, we can access the unqualified elements, in the xml document

no name-space

Note :- In the above we have used xsd as prefix, and this is recommended to be used in the Schema document. where prefix takes a unique name in the namespace.

=) will return back after we understanding, what we have in the body of the schema element.

=) Complex Type :-

Q) what is ComplexType?

Ans :- =) This allows to define a type having one or more childs and or attributes.

=) This type allows to define element types, that can be basic/ Simple / ComplexType.

for eg :-

```
<address>
<flatno> 786 </flatno>
<appartment> Mitrivihar </appartment>
<Street> ameerpeta </Street>
<State> ap </State>
<pin> 500037 </pin>
</address>
```

=) address in the above is a ComplexType, having 6 childs.

Q) How To create Complex Type?

Ans

```
<xsd : complexType  name= " ">
  <xsd : sequence>
        (or)
  <xsd : all>   (or)  <xsd : choose>

<xsd : element  attributes />
  - - - - - - - -
  </xsd : sequence> (or)
  </xsd : all>  (or)
  </xsd : choose>
  <xsd : attribute  attributes />
  - - - -
  </xsd : complexType>
```

<xsd : complexType> : this tag is used to create ComplexType.

<xsd : Sequence> : this tag describes that the elements should be used in the Same order how they are declared. Add even respective to their minOccurs and max Occurs.

<xsd : all> : this tag describes that the element Can be declared in any order. But their occurance is for Only 0 or 1 time

<xsd : element> This tag is used To declare an element having a name and a type, and we Can also give additional description like minoccurance and max occurance.

=) This tag takes the following attributes

<xsd : element <attributes>>

a) name : describes the name of the element in the xml docu

b) type :- describes the type of the element and this attribu
Optional. if we don't use this attribute, the body-
-Content of the element cannot be empty.

c) minOccurs :- to specify the min Occurance of an element
The value can be specified as any positive integer, even zero

d) max occurs :- to specify the max Occurance the value should
be >= min occurs.

Note :- If min occurs is 0 then max Occurance defaults to 1.
If max occurs is unlimited, we use the value "unbounded".

=) Now let us write a simple xsd, for the above discussed
address tag.

```
<xsd : ComplexType name = "addressType">
<xsd : sequence>
<xsd : element name= 'flatno' type= "xsd : integer" />
<xsd : element name = "apparment' type = 'xsd : string" />
<xsd : element name = "street" type = "xsd : string" />
<xsd : element name = "city" type = "xsd : string" />
<xsd : element name = 'state' type = 'xsd : string' />
<xsd : element name = 'pin' type = "xsd : integer" />
</xsd : sequence>
</xsd : ComplexType>
```

18/9/12

⇒ In the above Xsd, we have used

Xsd: sequence>, alternatively we can also use <Xsd:all>

⇒ If we use <xsd:all>, the elements can appear for one time, in any order.

⇒ we can also use <xsd:choice>, along with <xsd:sequence>, to choose between the elements, similar to what we have seen in DTD document

<!Element Cinema (hero| heroin)>

Note:- for specifying the occurance we can use the occurance indicators like ? + * , as we have in DTD, But in the schema we use the attributes minOccurs and maxOccurs, it is better discribeable compared to the occurance indicators.

⇒ Say in the above xsd document, we wanted to describe saying the pin element is optional, then in the xsd document we declare as

<xsd: element name = "pin" type = "xsd: integer"
minOccurs = "0" maxOccurs = "1"/>

⇒ --- <xsd: choice>

This element can be used inside <xsd: sequence> to select between the elements

for eg:-

```
<xsd:complexType name="addType">
  <xsd:sequence>
  <xsd:choice>
  <xsd:element name="flatno" type="xsd:string"/>
  <xsd:element name="houseno" type="xsd:string"/>
  </xsd:choice>
  <!.... other elements ...>
  </xsd:sequence>
  </xsd:complexType>
```

⇒ Now us let take a requirement like prepare an xsd and xml doc and validate them.

<u>xsd document</u> :-

```
<xsd:schema
  <xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNameSpace="http://www.nit.com"
   xmlns="http://www.nit.com"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

<xsd:complexType name="addressType">
  <xsd:sequence>
  <xsd:choice>
  <xsd:element name="flatno"
    type="xsd:string"/>
    <xsd:element name="houseno"
     type="xsd:string"/>
```

```
< xsd : element . name = "apparstment"
      type = "xsd : String" />
<xsd : element  name = "street"
      type = "xsd : String" />
<xsd : element  name = "city"  type = "xsd : String" />
<xsd : element  name = "state"  type = "xsd : String" />
<xsd : element  name = "pin"  type = "xsd : integer" />
  </xsd : sequence>
  </xsd : complexType>
  <xsd : element  name = "address"
      type = "addressType" />
  </xsd : schema>
```

<u>xml document</u>

```
<address  xmlns = "http://www.nit.com"
   xmlns : xsi = "http://www.w3.org/2001/XMLSchema - instance"
   xsi : schemalocation = "http://www.nit.com
    File : ///D:/xml/xsd/address.xsd" >
      <flat no> 500 </flat no>
       <apparstment> Saraswati Nilayam </apparstment>
        <street> hyderabad - ameerpeta </street>
        <city> hyderabad </city>
        <state> at </state>
        <pin> 500037 </pin>
    </address>
```

<u>Note</u> :- In the xml document, to associate with the xsd document we used an attribute named

xsi : schemalocation, this is into the namespace.

http://www.w3.org/2001/XMLSchema-instance

=) Now write an XMLSchemaValidations :- .java.

```
pk com.net.dom;
import javax.xml.passers.*;
import java.io.*;
public class xml schemavalidation
    {
    public static void main(String... S) Throws Exception
  {
  DOcumentBuilderfactory dbf = DOcumentBuilderfactory.newInstance
                                                        ();
        dbf.setNameSpace Aware(true);
        dbf.set Validating (true);
  dbf.setAttribute ("http://java.sun.com/xml/jaxp/properties/
    Schemalanguage", "http://www.w3.org/2001/XMLSchema");
  DOcumentBuilder db = dbf.newDOcumentBuilder();
   db.setErrorHandler (new MyErrorHandler());
   File InputStream fis = new file InputStream ("address-xm");
        db.parse (fis);
        S.O.Pln ("document passed");
  }
}
```

```
package com.nit.validation;

import javax.xml.parsers.*;
import java.io.*;
import org.xml.sax.*;

public class MyErrorHandler implements ErrorHandler
{
}
public void error(SAXParseException se) throws SAXException
{
S.o.pln (se.getLineNumber()+" " + se.getColumnNumber()+
        " " +se.getMessage());
}

public void warning(SAXParseException se) throws SAXException
{
  S.o.pln (se.getLineNumber()+" " +se.getColumnNumber()+
      " " + se.getMessage());
}
}
```

=) To Compile the application set the path for

=) C:\jdk 1.6.0 \bin

=) javac -d . *.java

=) java com.nit.validation.XMLSchemaValidation

=) Students responsibilities.

a) Change the order of the element in the xml document and dtd

b) Instead of sequence use <xsd:all>

c) check applying the minOccurs and maxOccurs attributes to element.

20/9/12

Simple Types:-

⇒ Are the Basic types added with Restrictions. Imagine we have a requirement, where we have to describe a string should have min 5 characters and max 20 characters.

⇒ To Describe these types of requirements we go for simple types.

⇒ The following Restrictions can be applied to the Built-in-types.

a) length.
b) min length
c) max length.
d) max Inclusive.
e) min Inclusive
f) max Exclusive
g) min Exclusive.
h) total Digits.
i) fraction Digits.
j) enumeration.
k) whitespace
l) pattern

Q) How to declare a Simple Type?

Ans

```
<SimpleType name=" ">
    <restriction base=" ">
        // restrictions
    </restriction>
</simple type.
```

⇒ Commencing understanding the above restrictions.

a) length:- This Restriction used to specify the length

for eg:-

```
<xsd: SimpleType  name = "CustomerId">
    <xsd: restriction  base= "xsd: integer"> // String is also Correct.
      <xsd: length  value= "5" />
      </xsd: restriction.
    </ xsd: SimpleType>
```

⇒) The above Simple type can be a base type for another Simple type here we have the possibility to modify the length, and if we don't wanted to allow this, additionally use another attribute to the length restriction i.e fixed.

```
<xsd: length value= "5"  fixed= "true" />
```

b) <u>min length</u> :- used to describe the min length

for eg:-

```
<xsd: SimpleType  name= "non-Empty-String">
    <xsd: restriction  base = "xsd: integer">
      <xsd: minlength  value = '3' />
    <xsd: restriction>
    <\xsd : SimpleType>
```

c) <u>max length</u> :- used to describe the max length.

for

```
<xsd: Simpletype  name = "input-to-form">
    <xsd: restriction  base = "xsd: integer">
      <xsd: max length  value= "15" />
    <\xsd: restriction>
    <\xsd: simpletype>
```

d) minInclusive :- used to describe the >=

```
<xsd: SimpleType name = "Twenty Thousand -or- more">
<restriction base = "xsd : integer">
<xsd : minInclusive value = "20000" />
</restriction>
</xsd: SimpleType>
```

e) minExclusive :- used to describes ">" greaterthan.

```
<xsd : SimpleType name= "more than - twenty greaterthan">
<xsd : restriction base= "xsd : integer">
<xsd : minExclusive value = "20000" />
</xsd : restriction>
</xsd : SimpleType>
```

f) maxInclusive :- used to describes the equals to <=

"lessthan-or- five thousand").

```
<xsd : SimpleType name = "Twenty thousand -or- lessthan">
<xsd : restriction base = "xsd : integer">.
<xsd : max Inclusive value = "50000" />
</xsd : restriction>
</xsd : SimpleType>.
```

g) max Exclusive :- used to describes < (lessthan)

```
<xsd : SimpleType name = "lessthan five thousand">
<xsd : restriction base = "xsd : integer">
<xsd : maxExclusive value = "5000" />.
</xsd : restriction>
</xsd : simpletype>
```

h) <u>totalDigits</u> :- describes the total no. of digits.

(i) <u>transaction digits</u> :- This restriction describes the digits after the decimal

```
<xsd: SimpleType    name= "roomTemperature">
  <xsd: restriction    base= "xsd: decimal">
    <xsd: totalDigits.    value= "4" />
    <xsd: fractionDigits    value= "1" />
    <xsd: minInclusive    value= "23.4" />
    <xsd: maxExclusive    value= "28.4" />
  </xsd: restriction>
</xsd: simpletype>
```

(j) <u>enumeration</u>:- describes set of values.

for eg:-

```
<xsd: SimpleType    name= "CurrencyType">
<xsd: restriction    base= "xsd: string">
  <xsd: enumeration    value = "INR" />
  <xsd:      "         "     = "USD" />
  </xsd: restriction>
</xsd: simpleType>
```

(k) <u>whitespace</u> Describe the whiteSpace to be replaced, removed or preserved.

for eg:-

```
<xsd: SimpleType    name= "Token">
<xsd: restriction    base= 'normalised String'>
  <xsd: whitespace    value= "collapse" />
  </xsd: restriction>
</xsd: simpletype>
```

⇒ this type restriction removes the leading and trailing white space.

⇒ If we don't want to remove the leading and trailing white spc use the values as preserve.

(f) pattern:- Describes the pattern; i.e an Regular Expression.

[abc] ⇒ Describes either a or b or c

[^abc] ⇒ " neither a or b or c

[a-d] ⇒ " we can use any character through a to d (lower cased)

[A-D] = through A to D [Upper "]

[a-d A-D] ⇒ either 'a' through 'd' or A through D.

[a-z [^bc]] ⇒ any char a through z expect b & c.

[a-j [0-z]] ⇒ a through j or '0' through 'z'.

\d = describes a digit b/w 0 to 9.

\D = " 0 not a digit (^0-9)

\s = " whitespace.

\S = " not a whiteSpace.

\w = " a word.

\W = " not a word.

We Can use indicator like

+ ⇒ describing 1 or more

* ⇒ 0 or more.

? ⇒ 0 or one.

{n} ⇒ minimum n times.

{n, m} ⇒ minimum n times, max n times.

```
<xsd: Simple Type      name= "initial">
  <xsd: restriction    base= "xsd: string">
    <xsd: pattern      value= "[a-z]" />
  </xsd: restriction>
</xsd: Simple Type>
```

21/9/12

Q) Requirement we are interested to write an xsd document for the following Requirement.

- Emps
  - → Emp (Attr :- Empid, managesid, working)
    - → Name (Inner element)
      - → first name.
      - → last name.
    - → Sal (Element with text as Content and Attribute).
    - → Designation.
      (Note :- Occurance of Emp is for 2 to 20)
  - → DEPT (Attr :- Deptno)
    - → Dname.

=) Before we implementing the above requirement let us understand about the types of elements (recall while following the DTD standards we have learnt the different types of elements like ChildOnly, TextOnly, mixed, Empty, Any)

=) Here already we have understood to declare SimpleType. Complex type, we wanted to understand additionally declaring inner elements, element with attribute and text, empty element mixed element and declaring attributes.

for Inner Elements (Inner types).

```
<xsd: element name = "Name")
  <xsd: ComplexType>
   <xsd: Sequence>
    <xsd: element name= "fn" type= "xsd: string" />
    <xsd: element name = "ln" type= "xsd: string" />
   </xsd: Sequence>
  </xsd: ComplexType>
</xsd: element>
```

=) To give additional info to the element Content are we attributes, we already know the different types of attributes, as per. the DTD standards additionally in Schema we can use. the built in types also.

Syntax:

```
<xsd: attribute name= " " type= " " use= " "
                default= " " fixed= " " >
```

=) The "use" of this element is used to described whether the attribute is requirement, optional or prohibited the default value is "optional".

=) The default attributes values of this element is any compatable type, for the declared attribute element.

for eg:-

```
<xsd: attribute name = "talk" type = "talktype"
                default = 'Super hit' >

<xsd: attribute name= 'fee" type= "xsd: double" fixed='4000'>
```

Note:- The Attribute element is declared inside the ComplexType element or inside <xsd: extension> element where <xsd: extension> element is a child element of <xsd: Simple Content>

=) where <xsd:SimpleContent> is child of <xsd: ComplexType>

=) This Sequence we use when we want to describe elements with attributes and Text Content.

```
<xsd:ComplexType name= " ">
<xsd: Simple Content>
<xsd: extension base= " ">
<xsd: attribute name= " " Type= " " default = " " />
</xsd: extension>
</xsd: SimpleContent>
</xsd: ComplexType>
```

=) Similarly we can declare empty elements and mixed elements where again Empty elements can have attributes, and Mixed elements contain text, as well as child elements.

```
< Customer Custid: "ci" />
< xsd: element name= "Customer">
    < xsd: ComplexType>
```

```
<xsd: attribute name="Custid" type="xsd: String"/>
</xsd: Complex type>
</xsd: element>
```

=) Similar for mixed element

```
<xsd: ComplexType mixed="true">
</xsd: ComplexType>
```

=) Now we have to create xsd for emps

emp-xsd

```
<xsd: schema xmlns:xsd="http://www.w3.org/2001/XMLSchema
        TargetNameSpace="http://www.nit.com"
        xmlns="http://www.nit.com"
        elementformDefault="qualified"
        attributeformDefault="unqualified">

<xsd: SimpleType name="workingType">
<xsd: restriction base="xsd: String">
<xsd: enumeration value="yes"/>
<xsd:      "      value="no"/>
</xsd: restriction>
</xsd: SimpleType>
<xsd: SimpleType name="CurrencyType">
<xsd: restriction base="xsd: String">
<xsd: enumeration value="INR"/>
<xsd:      "      value="USD"/>
</xsd: restriction>  </xsd: SimpleType>
```

```
<xsd: ComplexType name= "salType">
<xsd: SimpleContent>
<xsd: extension base= "xsd: double>
<xsd: attribute name= "ostype" type= "CurrencyType"
    default= "NA"/>
  </xsd: extension>
  </xsd: SimpleContent>
  </xsd: ComplexType>

<xsd: ComplexType name= "empType">
 <xsd: Sequence>
   <xsd: element name = "name">
    <xsd: ComplexType>
    <xsd: Sequence>
   <xsd: element name= "fn" type= "xsd: String"/>
   <    "     "    "  = "ln"  "     "     />

   </xsd: Sequence>
   </xsd: ComplexType>

</xsd: element>
<xsd: element name= "sal" type= "salType"/>
<xsd: element name= "designation" type= "xsd: String"/>
</xsd: Sequence>
<xsd: attribute name= "empid" type= "xsd: ID" use= "required"/>
<    "     "     "  = "managerId" type= "IDREF" " = "optional"/>
<    "     "     "  = "working" type= "workingType" default= "yes"/>
</xsd: ComplexType>.
```

```
<xsd: ComplexType name = "dept type">
<xsd: Sequence>
<xsd: element name= "dname" Type= "xsd:String" />
  </xsd: Sequence>
<xsd = attribute name= "deptno" type= "xsd:ID" use = 'requrie
</xsd: ComplexType>
<xsd: ComplexType name= "empstype">
<xsd: sequence>
<xsd: element name= "emp" type= "empType" minOccurs= "2" maxOccurs
                                                          = "20" />
<xsd: element name = "dept" type= "deptType" />
</xsd: sequence>
</xsd : Complex Type>
<xsd : element name = 'emps" type= "empstype" />
</xsd: Schema>
```

22/10/12

Xml document

~~xsd sequence~~

```
<emps xmlns = "http://www.nit.com"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation = "http://www.nit.com file:///D:/xml/xsd1
    /emp.xsd">

<emp empid="e101" managerid="e102" working="yes">
    <name>
        <firstname> naveshi </firstname>
        <lastname> Technologies </lastname>
    </name>
    <sal cotype="INR">10000</sal>
    <designation> developer </designation>
</emp>
<emp empid="e102">
    <name>
        <firstname> nit </firstname>
        <lastname> technologies </lastname>
    </name>
    <sal cotype="USD">10000</sal>
    <designation> TM </designation>
</emp>
<dept deptno="d10">
    <dname> deparsn development </dname>
</dept>
</emps>
```

=) Now validate the emp.xml and emp.xsd, document and read data from the xml document.

=) requirement is to read all the Salary element value.

=) In previous example also we have validated, the same lines of code, and same Errorhandler only in the XmlSchemaValidation.

write the following code to "read" the Salary element value

```
package com.nit.parsers*;
-------------------------
-------------------------

Document doc = db.parse(fis);
        S.o.pln("document parsed");
Element emps = doc.getDocumentElement();
NodeList emps_childs = emps.getChildNodes();
for(int i=0; i<emps_childs.getLength(); i++)
{
        Node emps_child = emps_childs.item(i);
if (emps_child.getNodeType() == Node.ELEMENT_NODE)
  {
  String element_name = emps_child.getNodeName();
      if (element_name.equals("emps"))
  {
        Element emp = (Element)emps_child;
NodeList emp_childs = emp.getElementsByTagName("sal");
        Node emp_child = emp_childs.item(0);
NodeList sal_childs = emp_child.getChildNodes();
```

```
Node sal_child = sal_childs.item(0);
        S.o.pln (sal_child.getNodeValue());
    } } }
  } }
}
```

⇒ Now validate the xsd and xml document using the Saxparser, do the following modification to the Saxparser example which we have written, earlier i.e into

⇒ SAXParserReadSample.java, include the following lines of the code.

```
factory.setNamespaceAware(true);
factory.setValidating(true);
parser.setProperty("http://java.sun.com/xml/jaxp/properties/
    SchemaLanguage", "http://www.w3.org/2001/XMLSchema");

    //remaining lines are same.
    // even the listener Application is Same.
```

⇒ Note:- for the Cinema.xml write a Cinema.xsd, validate xml and xsd documents and perform insert Operation.

⇒ jaxb.

⇒ AJAX

⇒ Webservices.

24/9/12

JAXB

=) Java Architecture for XML Binding

=) It is a Service API given into Java EE Specification.

=) JAXB provides an High level abstraction for XML Based Serialization i.e Converting a Java object into an XML document and even Converting XML document into a Java Object.

=) In short we say this is an utility for integrating XML to java application.

=) Generate Java classes for the emps.xsd document, use this document written in previous class,

=) A Command to generate is.

```
> xjc -p com.nit.jaxb emps.xsd
```

=) To execute this Command no additional approach (or) class path, setting have to be done if we are installed with jdk16 (or) above.

=) The following java classes gets generated like.

Empstype.java.
Emptype.java
Depttype.java.
Weringtype.java
package-info.java
Objectfactory.java

=) Now to perform marshalling i.e Converting java objs to xml document.

=) Steps are

1) Create the objects.
2) Get the JAXB Context type.
3) Get the Marshaller.
4) Result To marshal.

Sample example program for marshalling

```
package com.nit.jaxb;
import javax.xml.bind.*;
import java.io.*;
import java.util.*;

public class MarshallingTest.
    psvm(String...s) throws Exception
{
    //instantiate the Empstyle.

Empstype es = new Empstype();
    // get the list

List<Emptype> list = es.getEmp();
Emptype emp = new Emptype();
    emp.setEmpid("e101");
Emptype.Name name = new Emptype.Name();

    name.setFirstName("abc");
    name.setLastName("xyz");
```

```
emp.setName(name);
emp.setSal(1000.0);
emp.setDesignation("developer);
list.add(emp);

emp = new EmpType();
emp.setEmpid("102");
name = new EmpType.Name();
name.setFirstname("aaa");
name.setLastname("bbb");
emp.setName(name);
emp.setSal(1000.0);
emp.setDesignation("developer");
    list.add(emp);

JAXBContext jb = JAXBContext.newInstance("com.nit.jaxb");
    Marshaller m = jb.CreateMarshaller();
    m.marshal(new ObjectFactory().CreateEmps(es), new
        fileWrite("my emp.xml"));

    m.marshal(new ObjectFactory().CreateEmps(es), new FileWriter(
        ("my emp.xml"));
}  }
```

⇒ The next step is unmarshalling.

a) get the JAXBContext type
b) get the Unmarshaller
c) Result To Unmarshall

25/10/12

=) In the above example, we have transformed java classes in to an xml document.

=) Now the requirement to convert the xml document into corresponding set of Java Objects which we have termed as UnMarshalling.

=) The previous example we termed as Marshalling

```
package com.nit.jaxb;
import javax.xml.bind.*;
import java.io.*;
import java.util.*;

public class UnMarshallingTest
{
    psvm(String... s) throws Exception
{
```

previous example.

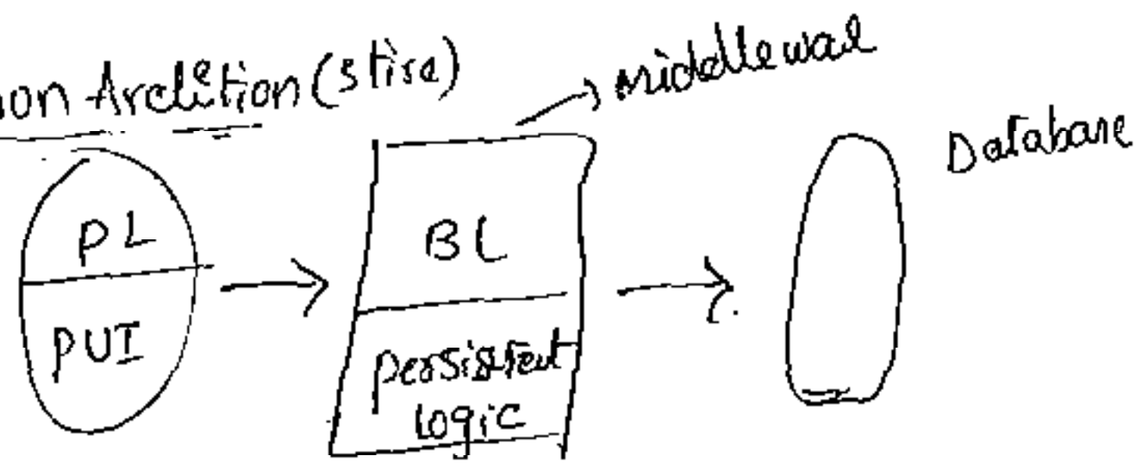
```
JAXBContext jb = JAXBContext.newInstance("com.nit.jaxb");
Unmarshaller um = jb.createUnmarshaller();
// get the JaxB Representation of xml element
JAXBElement<EmpType> je = (JAXBElement<EmpType>) um.unmarshal("new.java.io.File("myemp.xml")");
```

```
EmpStype et = je.getValue();
List <Emptye> emp = et.getEmp();
    //use enhanced for loop
for (Emptype e:emp)
{
    S.o.pln (e.empid);
    S.o.pln (e.getName().getFirstname()
    S.o.pln (e.getName().getLastname())
    S.o.pln (e.getSal());
    S.o.pln (e.getDesignation());
    } // for.
  } // class.
} // main.
```

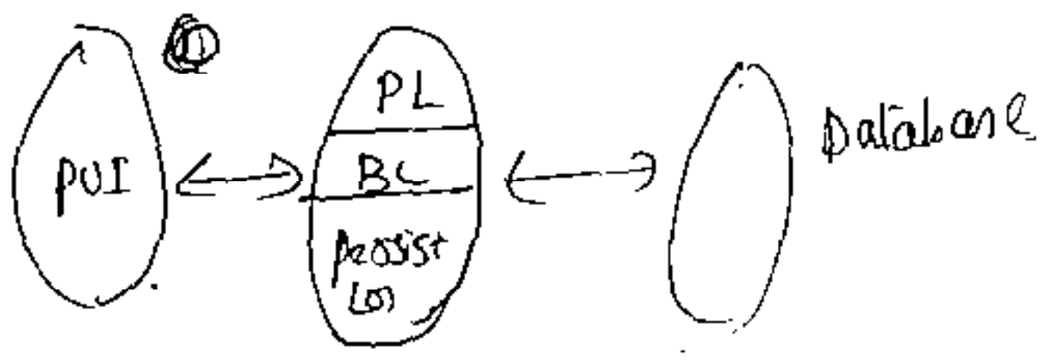
Save as => UnMarshallingTest.java.

Notes:

Tradition Architecture (3tire)



Model-3



web application (3tire).

web 2.0.

Rearrange "presentation logic" from middlee. ware to presentation layer. => It is web2.0

25/10/12

AJAX

(Asynchronous JavaScript and XML)

=) AJAX is a platform Supporting implementing web 2.0 Applicati

=) AJAX includes technologies like xml, javaScript for implementing infrastructures, for web 2.0 applications.

Note:- AJax is not limited to java app. only, we can use it with Technologies like ASP, .NET, PHP.

Q) What is web 2.0?

It is an enhancement to web Architecture.

Understanding web 2.0 architecture.

=> This Architecture Supports executing the presentation logic in the client tier.

Note:- we have engines likes javascript, that Supports the above requirements, for the additional requirements, using javascript language increases the Complexity, Solutions were Applet's and Adobeflash, But these have their respective limitations. like with respect to Security.

=) web 2.0 architecture reduces the network traffic.

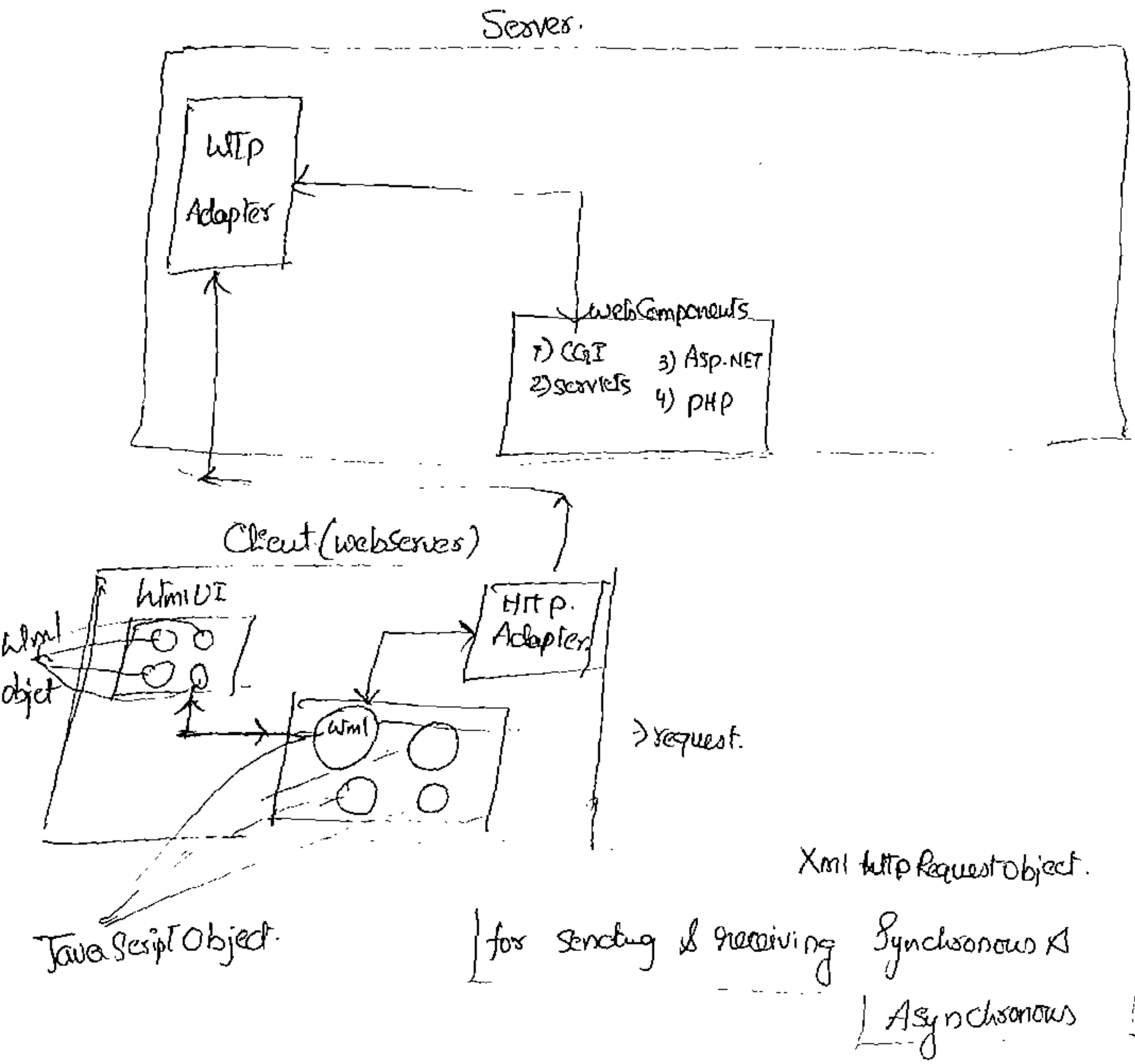
=) web 2.0 Supports to implement Rich UI

=) So Google has identified a platform, Combined with technologies like xml, javascript and HTTP and has given the name AJAX.

=) An it has used AJAX in its application like Gmail, GoogleMaps.

=) AJAX also has many limitations, as it is using javascript and it is difficult to debug javascript.

=) Apart from AJAX we also have Technologies that are motivited from AJAX, They are Adobe flex and SilverLight and JavaFx. With Javafx we have performance issues, and SilverLight is restricted to i.e



Note : XMLHTTP Object is used for receiving & Sending Synchronous and Asynchronous messages.

--- The Object Supports functions like

Open() // this behaviour is Overloaded
Send()

properties like
onreadystatechange
readyState
responseText
response XML

=) Now let us write a Simple example and.

```
<html>
<head>
<script>
 var request;
 function sendRequest(tf) {
                 //instantiate XMLHttpRequest
        request = new XMLHttpRequest();
   //   (request = new ActiveXObject ("Microsoft.HttpXML");
                                          ↳ Only for Internet Explore
 request.onreadystatechange = processRequest;
 request.open ("GET", "http://localhost:7000/ex1/Test.jsp?
        uname = + tf.value);
      request.send(null);
 }
```

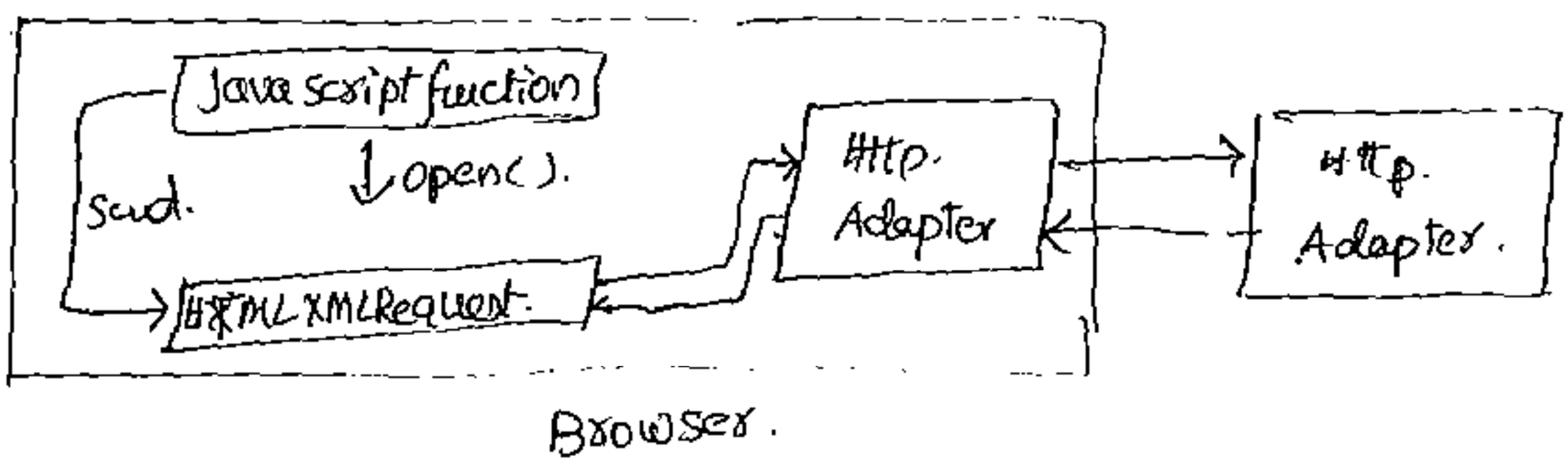
```
function processRequest()
{
  if (request.readyState != 4)
{
  mess = "Loading ... ";
}
 else if (request.readyState == 4)
{
      mess = request.responseText;
  }
document.getElementById('resp').innerHTML = mess;
  }
    </script>
    </head>
    <body>
First Ajax Application
        Name <input type = "text" onblur = "SendRequest
                                        (this);" />

<input type = "button" value = "check" />
  <div id = "resp"> Respons will Come here </div>

        </body>
        </html>
```

Save:- Tomcat ⇒ [Ajax] ⇒ { Home-Wml ⇒ Currentapp.
folder WEB-xml ⇒ WEB-INF
Text.jsp

test.jsp

welcome by <% out.println (request.getParameter ("uname"));
%> To Ajax.

27/9/12



Understanding XMLHttpRequest Object.

⇒ It is an javascript Ajax object for sending HttpRequest & getting HttpResponse in a Synchronous or asymnous manner.

⇒ In above application we created their object,

request = new XMLHttpRequest();

// for i.e
request = new ActiveObject ('Microsoft.XMLHTTp');

// for browser dependency we can write following code.

```
function createRequest()
{
    try {
        request = new XMLHttpRequest();
    }
```

```
Catch e
  {
 try
   {
    request = new ActiveObject ('MSXML2.XMLHTTP');
   }
 Catch e.
  {   Try {
        request = new ActiveObject ('Microsoft.XMLHTTP');
        }
 Catch (e)
 {  center ("may be ur browser problem");
    return false;
 }
  } }.
```

// Similarly we can also have if condition to describe the above

request.onreadyStateChange = callback.

when the readyState is change this call back is invoked.

The XMLHttpRequest Object falls into the following state.

0 - uninitialized
1 - Connection established.
2 - request sent.
3 - request processed.
4 - request response completed.

where this object supplies the following 5 properties.

a) alreadyStatechange
b) readyChange
c) responseText d) Status.

⇒ In our Example sit the can be as property to.

can ready state change

request.alreadystatechange = processRequest;

⇒ This Object Supports ~~fotir~~ following 3 behaviours

(a) Open.
(b) SetRequestHeader (optional)
(c) Send.

Openbehaviour is overloaded.

Open (request-method, url)
Open (request-method, url, boolean):- In this behaviour boolean describes request is Synchronous or Asynchronous and default to asynchronous.

Open(request-method, url, boolean, b, Usersname)

Open (request-method, url, boolean, a, username, password)

⇒ username & password describes, the username & password for accessing the web resources.

⇒ Send (process) — In above example we wrote this five. as the request method is GET, there is no body Content.

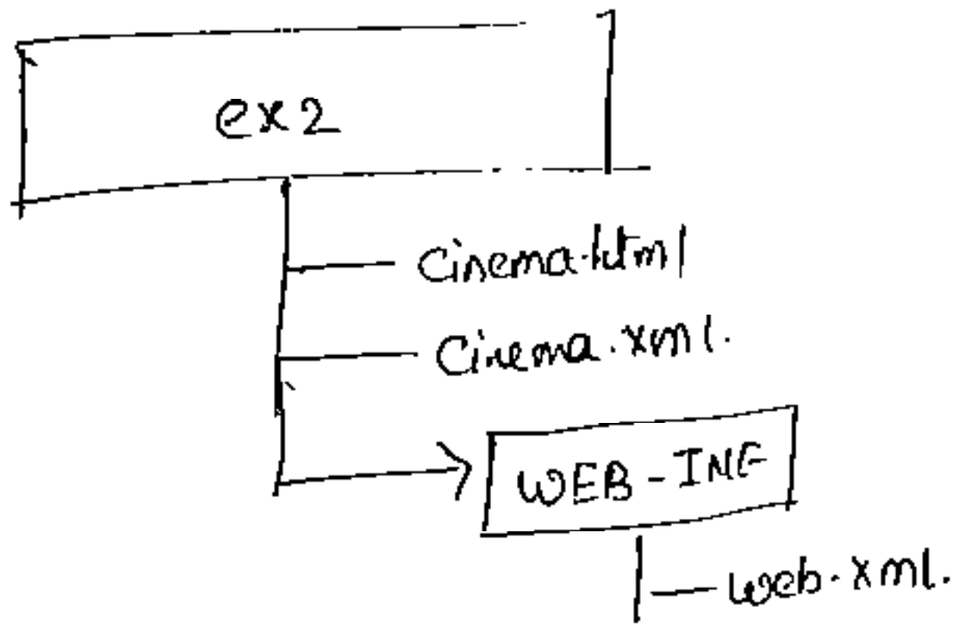
⇒ if the request method is Post, there ~~is~~ will be body Content. When the behaviour is invoked a new thread.Object is created.

28/9/12

Understanding java Script dom parser.

```
<html>
<head>
<script>
        function getCinemas() {
  Var domparser = new ActiveObject("Microsoft.XMLDOM");
  Var doc = domparser.load("Cinema.xml");
  Var Cinemas = domparser.documentElement;
  Var Cinemas_childs = Cinemas.ChildNodes;
  Var op = document.getElementById("result");
  for(var i=0; i<Cinemas_childs.length; i++)
  {
Var cinemas = domparser.documentElement;
  Var Cinemas_childs = Cinemas.ChildNode;
  Var op = document.getElementById("result");
  for(var i=0; i<cinemas_childs.length; i++)
{
      var Cinemas_child = cinemas_childs.item(i);
  Var
Cinema_childs = cinema_child.getElementsByTagName("name");
  Var Cinema_child = cinema_childs.item(0);
  Var name_childs = cinema_child.ChildNodes.item(0)
      .nodeValue;
      op.innerHTML += name_childs + ", "
  }
}
  </script>
  </head>
  </body>
<input type="button" value="load xml" onClick="getCinemas()"/>
```

```
<div id="result"> Cinema Names </div>
</body>
</html>
```

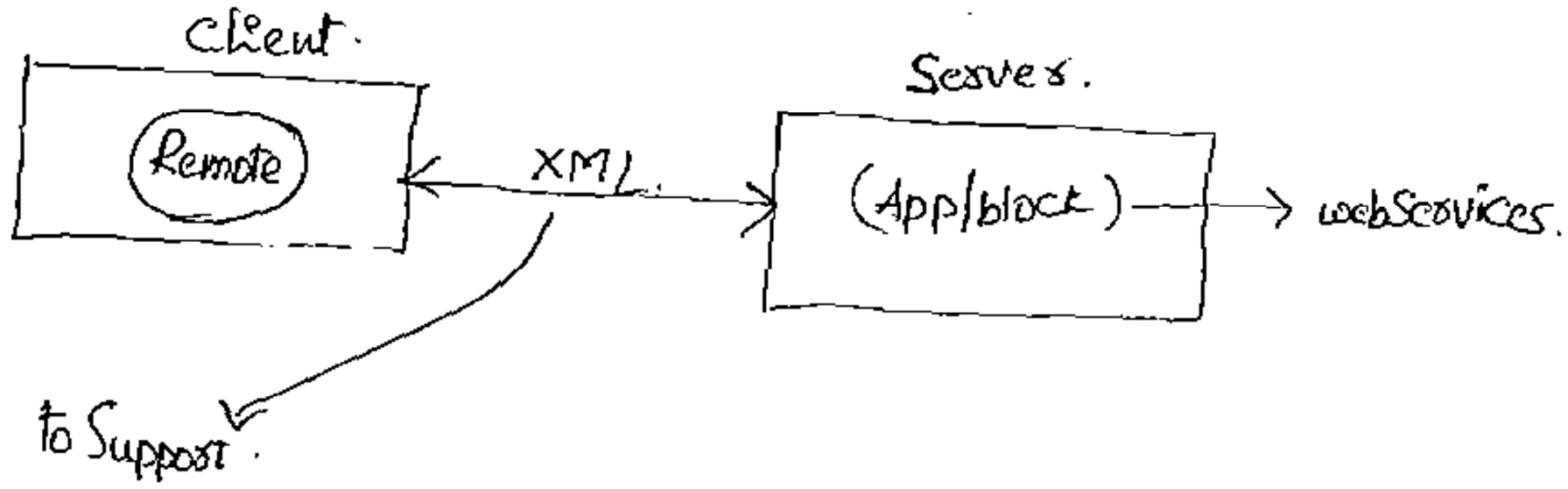


2/10/12

Web Service

Web Service :-

=) It is an application or block of (executable) code exposed to the clients (Remote) through XML based protocols.



Heterogeneous distributed System

Note:- webServices initially were designed To Service web Clients as the web Clients were intelligent to (understand) use the Transportation protocol i.e HTTP.

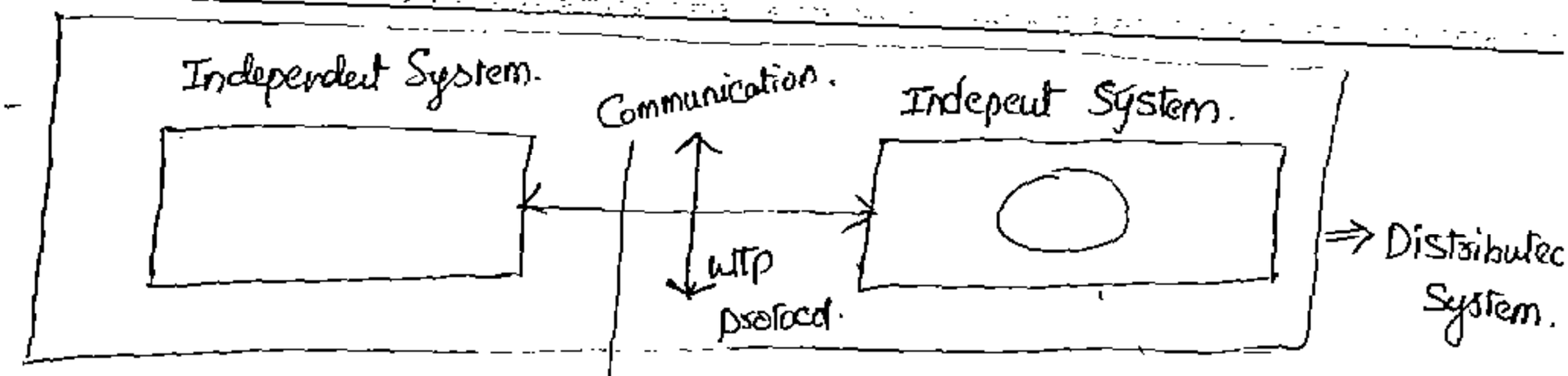
Now a days we have Advancements in webServices, so we can use any transport guaranteed protocol for transporting the messages. Even HTTP or any transport guaranteed protocol.

=) understanding the definition.

Distributed System :-

It is a collection of independent Systems Communicating with one another by exchanging messages.

Note :-



Indepent Sys. Communicating one another by exchange messages. is known Distributed System.

Message Communication problem.

→ should be like Indepents

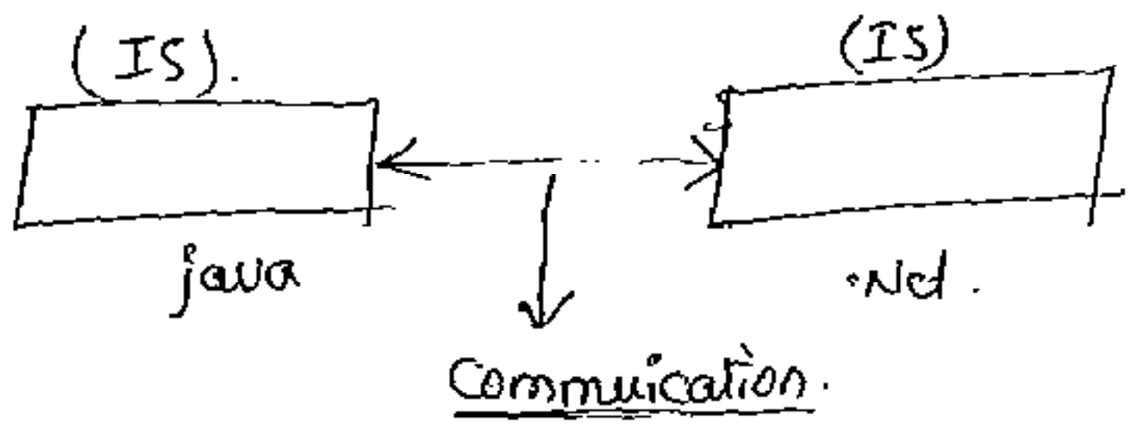
Vendor Indepent ⇒ RMI

JRMP protocol ⇒ RMI [only Java language]

8686169871.
Sitaram
9885516686
9160580994

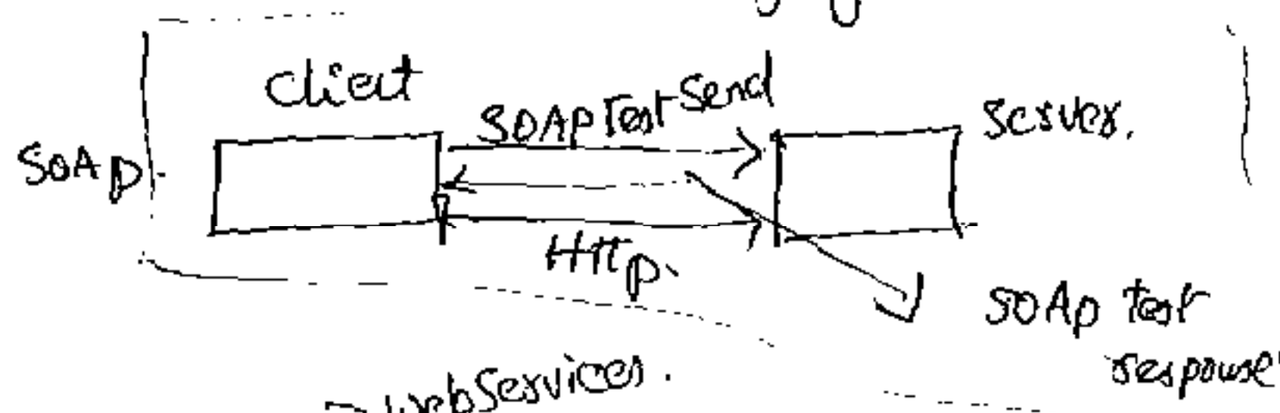
⇒. Only Communication to another they have own disadvantages [because it is idependency] to make it is "dependency"

(*) RMI Communication. leades the network problem, presently not used.



[that only java - java] not java - (net) or php

* java, .Net, php platform are accepted Xml language because.
  * Text base, (xml).
  * extensible.



WebServices:-

1) xml protocol
2) dynamic location (find out) (Infrastructure reqd) → WebServices.
3) Creating (Java Infrastruct and Integrating with WebServices)
4) accessing.

=) So here we have various problems, and we were given the various Solutions, inform of various Technologies, which we now list them out.

=) problem was wrt Design and implementing the messaging and transportation protocols.

=) Solutions were technologies like.

RMI :- for java 2 java Communication

=) for messaging it uses java Serialization.

=) for transportation it uses JRMP.

(for details refer javaEE notes)

=) But the requirement is Heterogenous (java, .Net, php, c#...) Distributed System. So here we required a Unified messaging protocol. This is where we find the need of Xml provides benfits like.

a) It is self descriptive.

b) It is text based.

c) It is extensible.

d) widely accepted by various Communication like.

— java, c#, php, .Net

In webservices.

Note:- Creating message and dealing the message (Comm-message) is the main aim. Not use the to Concentrate abt transportation http protocol.

From the above discussion, we now say

webservice is designed to support communication b/ independent systems implemented in different language and are into different platform.

=). Integrating the business collaboration (merging 2 business service

=) Understanding the Architecture of webservice :

From the discussion above we understood using xml is a wise solution, for messaging, but to understand the arch, we have to understand the following elements

a) Unified xml protocol
b) Standards describing the web services
c) Infrastructures to dynamically locate and access the web service.

note: And a standardized protocol for transporation.

note: Xml describes the request and response messages understandable to different clients. It is simple and human readable.

SoAP

(Simple Object Access protocol)

Note: But from SoAp 1.2 onwards we don't abbrivate SoAp as above

⇒ SoAp is an XML ML, describing the Messages.

⇒ for Better InterOperability (operability b/w Systems) we have an organization named WS-I (WebService InterOperability), which includes Various Venders and other Communities.

⇒ WS-I provides platform for creating and accessing the webservices, which is only termed as 'Basic Profile' its versions are 1.1, 1.2.0

⇒ SoAp is one of the element in To BP-I (platform) (Basic profile)

⇒ Services Comptable with BP 1, and Can send and receive text msg. using SoAp protocol.

⇒ SoAp protocol is Can be embedded on the top of Http.

⇒ where the SoAp request msg describes the type of operation and i/p types and the soap response describes the operation is Successful or a fault.

⇒ SoAp allows wide variety of clients accessing the webservices. The client Can access the webservice only when it is described to the client. Now our responsibility is to understand how to describe webservice to the clients.

WSDL (Web Service Description Language):

⇒ It is an XML ML describing the webservices to the Clients in a language neutral manner.

⇒ So ~~webservice~~ the clients can go for accessing the webservices, only after getting the WSDL document.
↳ client understan webservices logic

⇒ version of WSDL are 1.0, 1.1, 1.2

⇒ The WSDL 1.2 is renamed to WSDL2.

⇒ WSDL is also into BP1

⇒ Now we are interested to locate the webservice dynamically.

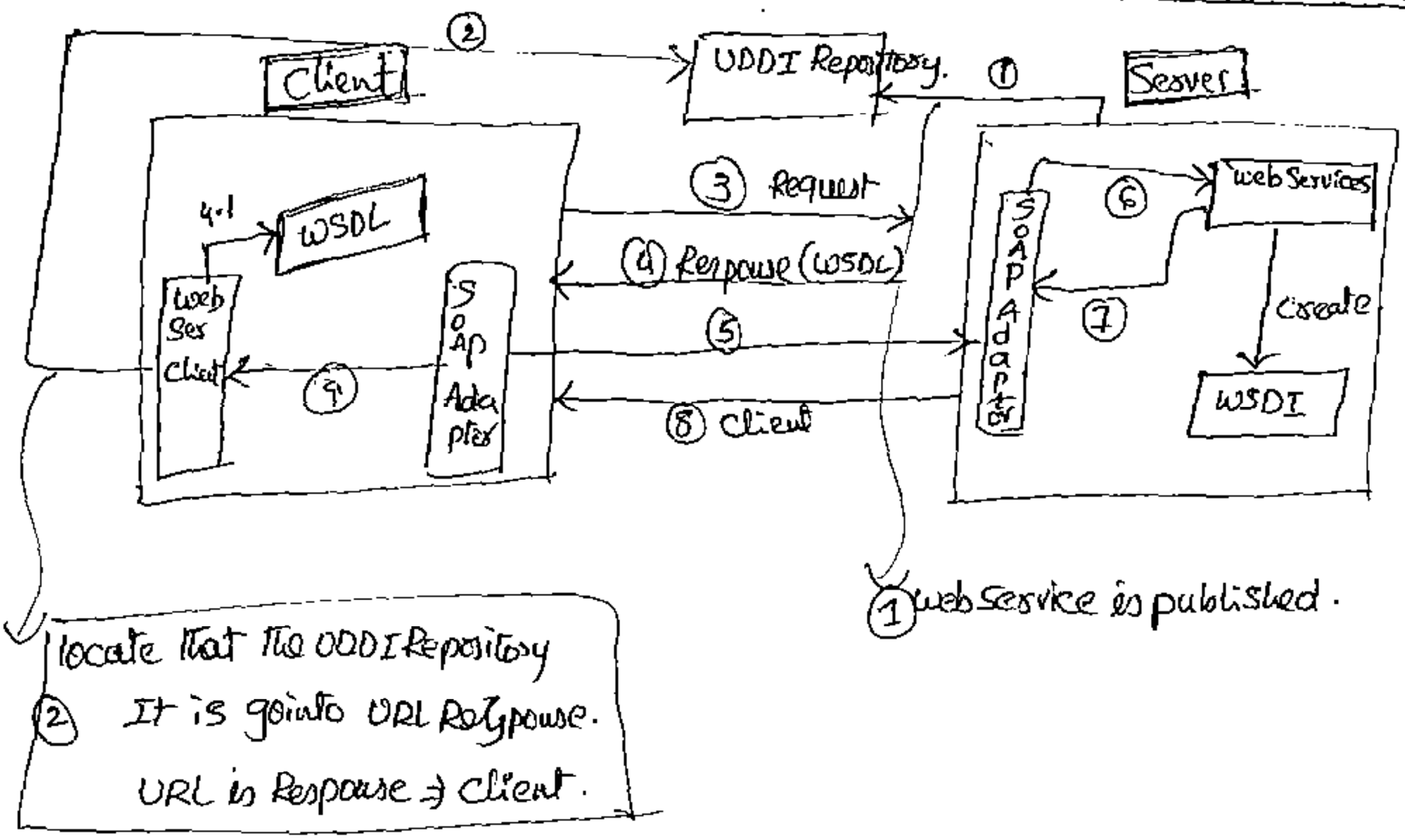
⇒ So that webservices can be reachable to more no. of clients So here we require a repository type, where the webservices can be published, which should be again as XML based, where the clients and access it using xml protocols.

UDDI (Universal Description Discovery and Integration)

⇒ UDDI is an XML based registry, used to public web services. which is accessed using xml protocols.

⇒ putting All the three into an arch, we form the webservice arch.

Note:- Service wants to expose more no. of clients UDDI Registery mandatory.
for Single client only WSDL Sufficient.



locate that the UDDI Repository
(2) It is gointo URL Response.
URL is Response ⇒ client.

(3) URL Resquest [client - server]

(4) Response [server-client] [what is webservice it is exposed]

(4.1) Client read (WSDL) and make SOAP request.

(5) SOAP Request. [client (SOAP Adapter) to Server (SOAP Adapter).]

(6) SOAP Request [SOAP Adaptor to web services.

(7) Response [webservices to SOAP Adaptor].

(8) SOAP Response. [SOAP Server to SOAP Client]

(9) Resolve the this [SOAP Response.] give to the webserver client.

WEB - SERVICES Architecture

4/10/12

webService and Java

As Now a days, most of the Applications are Enterprise applic-tions, i.e the applications Composed of Small and large Compon providing Software Solutions to business requirements, and we use webservices in these type of application Development. So into java platform also we have no. of infrastructures for Creating and accessing the webservices, Creating 'WSDL' Creating and resolving SOAP messages.

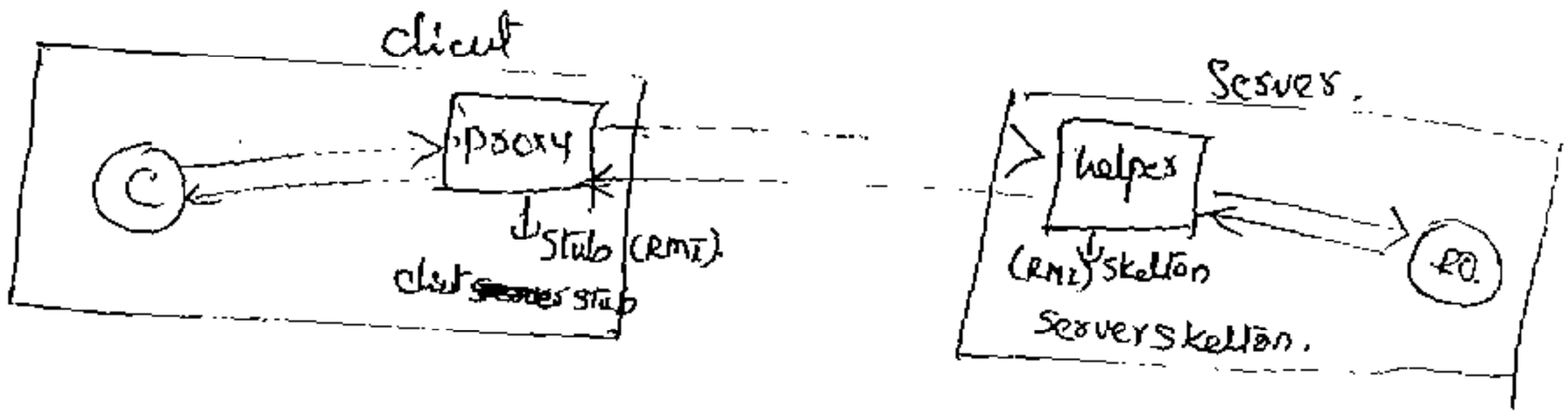
⇒ we now list out the API'S into javaEE platform, which provides infrastructures for webservices Creation and access.

(Integrated set of individual API for simply the Business Servic

a) jax-rpc
b) saaj
c) jaxm
d) jaxr
e) jaxb
f) jax-ws (new from javaee5)
g) jax-rs

Jax-rpc :-

(Java api for xml based remote procedure call)

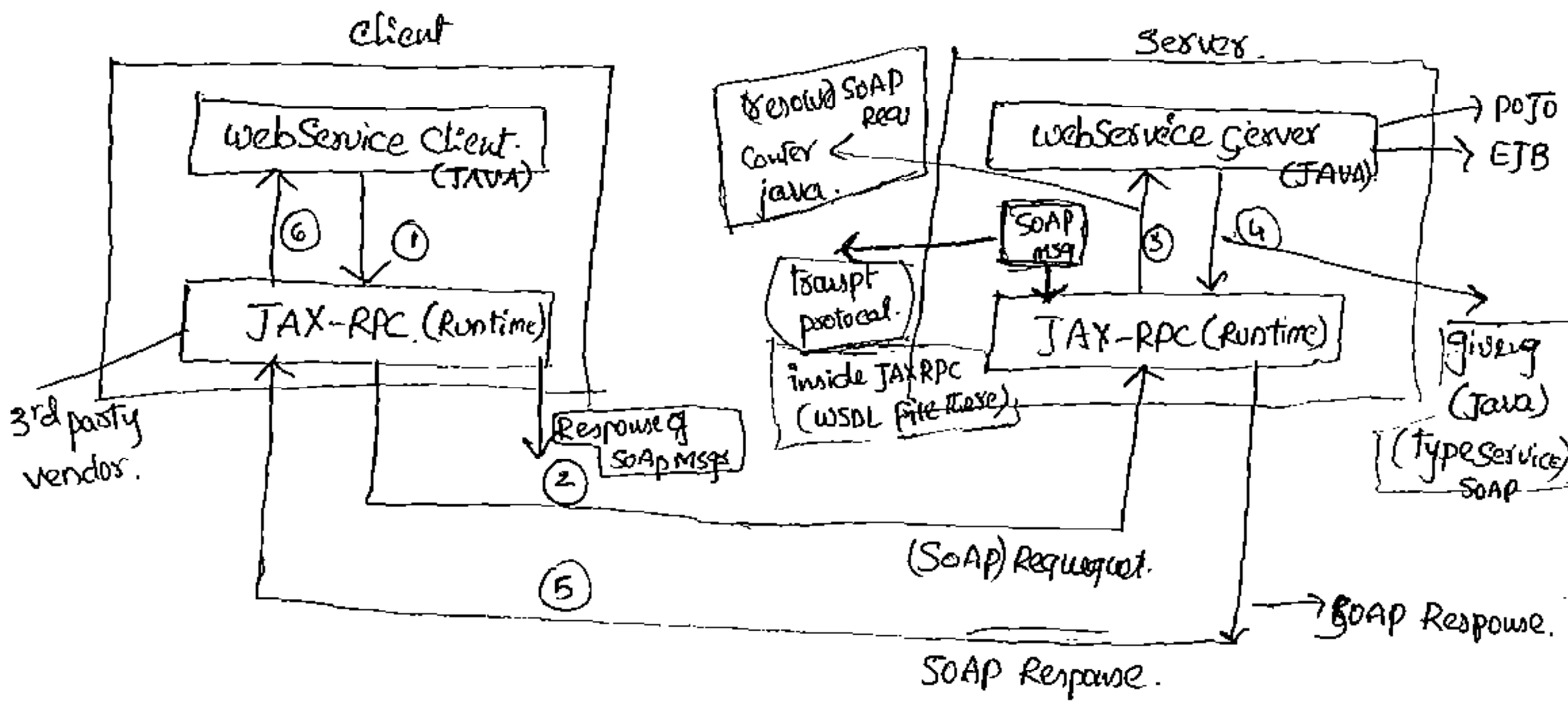


5/10/12

⇒ JAX-RPC is one of the Service API's in To JavEE Specification.

⇒ JAX-RPC provides an high level abstraction for exposing and accessing the webServices.

⇒ JAX-RPC uses Rpc style of implementing the Infrastructure which is categorized into 3 architectures.

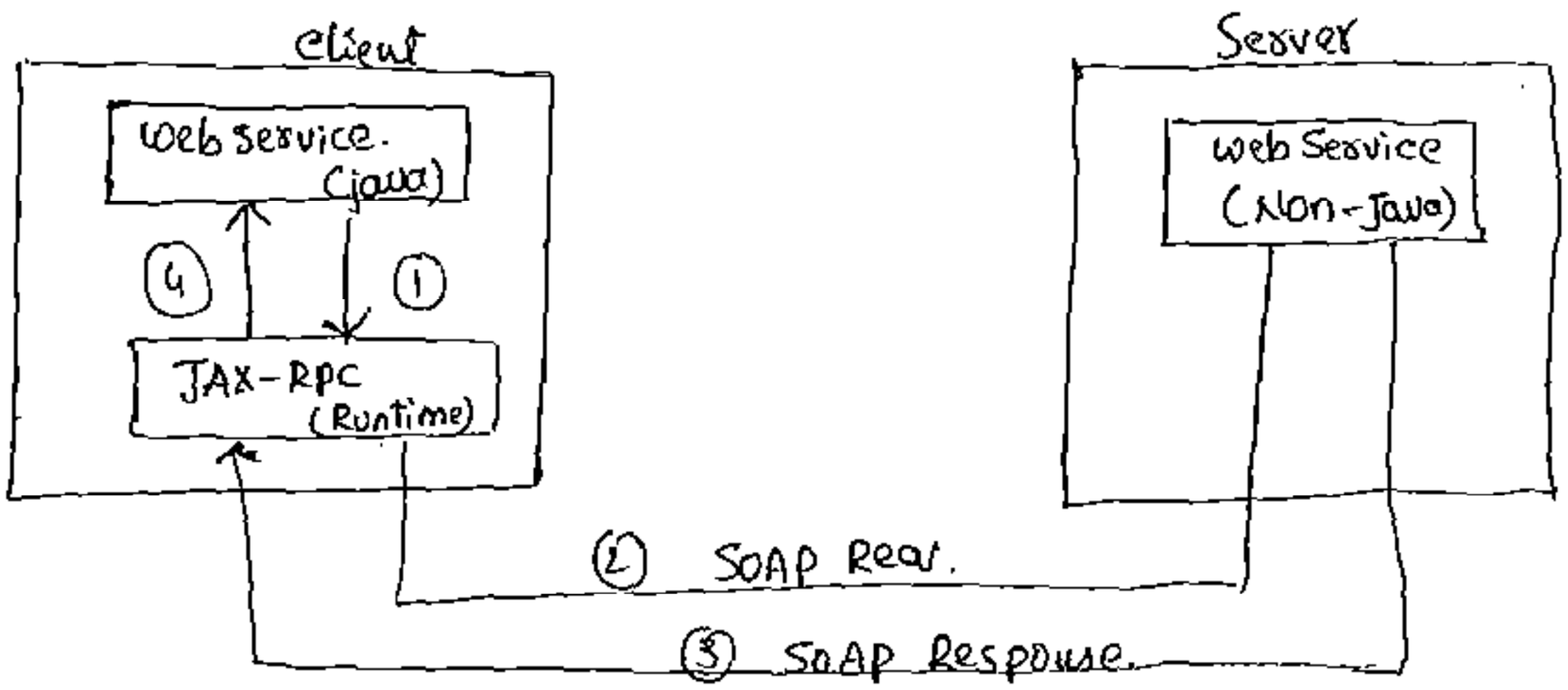


⇒ The above architecture describes the Client – Service arc into java, where the Service can be a POJO or an EJB.

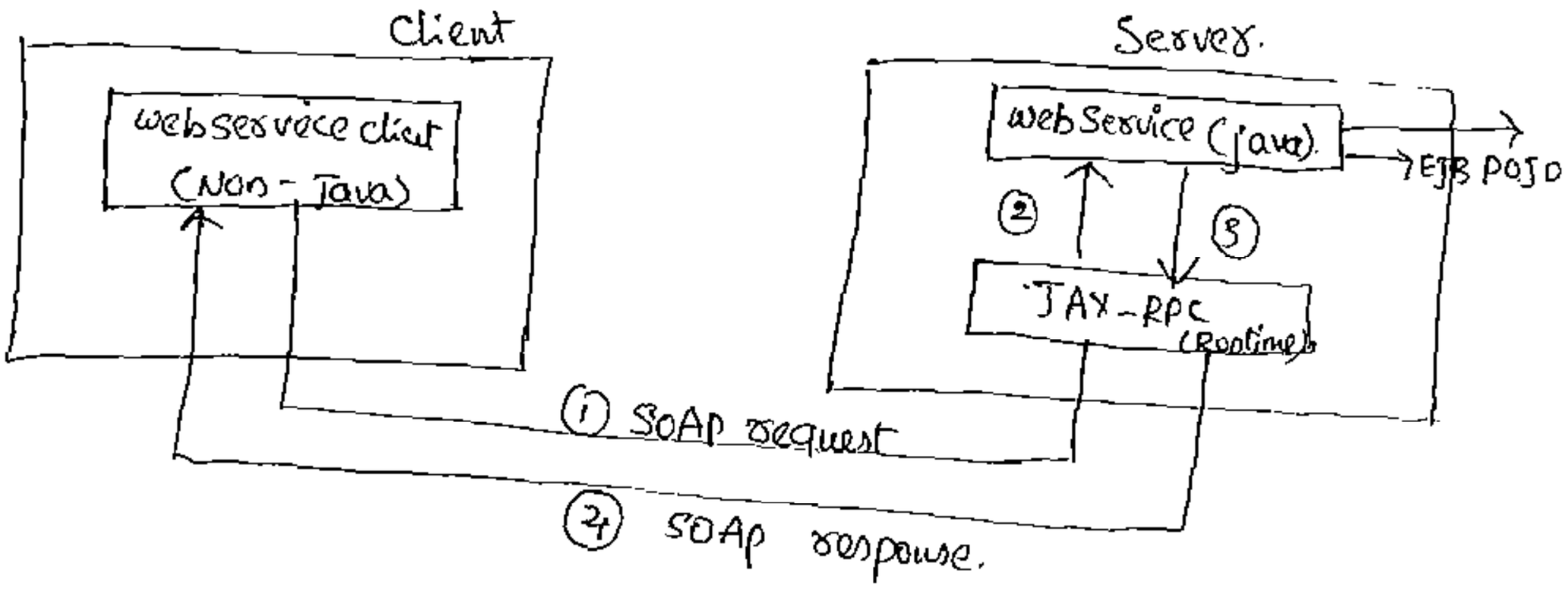
⇒ The JAX-RPC runtime is an Convinence or Infrastrutre implemented by 3 party vendors, where the webClient Communicates with JAX-RPC runtime, and JAX-RPC runtime already has WSDL document, prepare a SOAP request and send it To Server Side.

The Server side JAX-RPC runtime resolves the SOAP req, Convert & gives the request to web Service application. where the web-services application processes the request, & gives the respons to JAX-RPC runtime. JAX-RPC runtime prepares the SOAP response & transports it to the client. The client side JAX-RPC resolves the SOAP respouse & gives the respouse to the webService client.

⇒ If the client is Java & Service is Non-java (or Java without JAX-RPC)



⇒ If the Client is Non-Java (or java without JAX-RPC runtime), and Service is into Java



⇒ JAX-RPC is the Specification, now we should go implementation of JAX-RPC.

Note: J2SE (Sun Specification) JDK (Sun (dev) implementation) (OpenSource) lik 3 vendor

=) Now, we commence the learning & understanding implementation of JAX-RPC, where we have various implementations given by 3rd party vendors like Sunmicrosystem, Apache.

=) we commence our learning understanding AXIS implementation given by Apache.

=) Axis provide infrastructure to resolve the SOAP request to delegate to the webservice for operations.

=) Axis can be deployed into any J2EE or JavaEE compliant application servers like websphere, weblogic Sumon servers, pramati (Hyd based server).

=) Axis Implementation in the server is a servlet i.e Axis Servlet, so it can be deployed into any java web application server.

=) Axis supports pojo and EJB

=) To ~~display~~ deploy the webservice into Axis platform we have two styles.

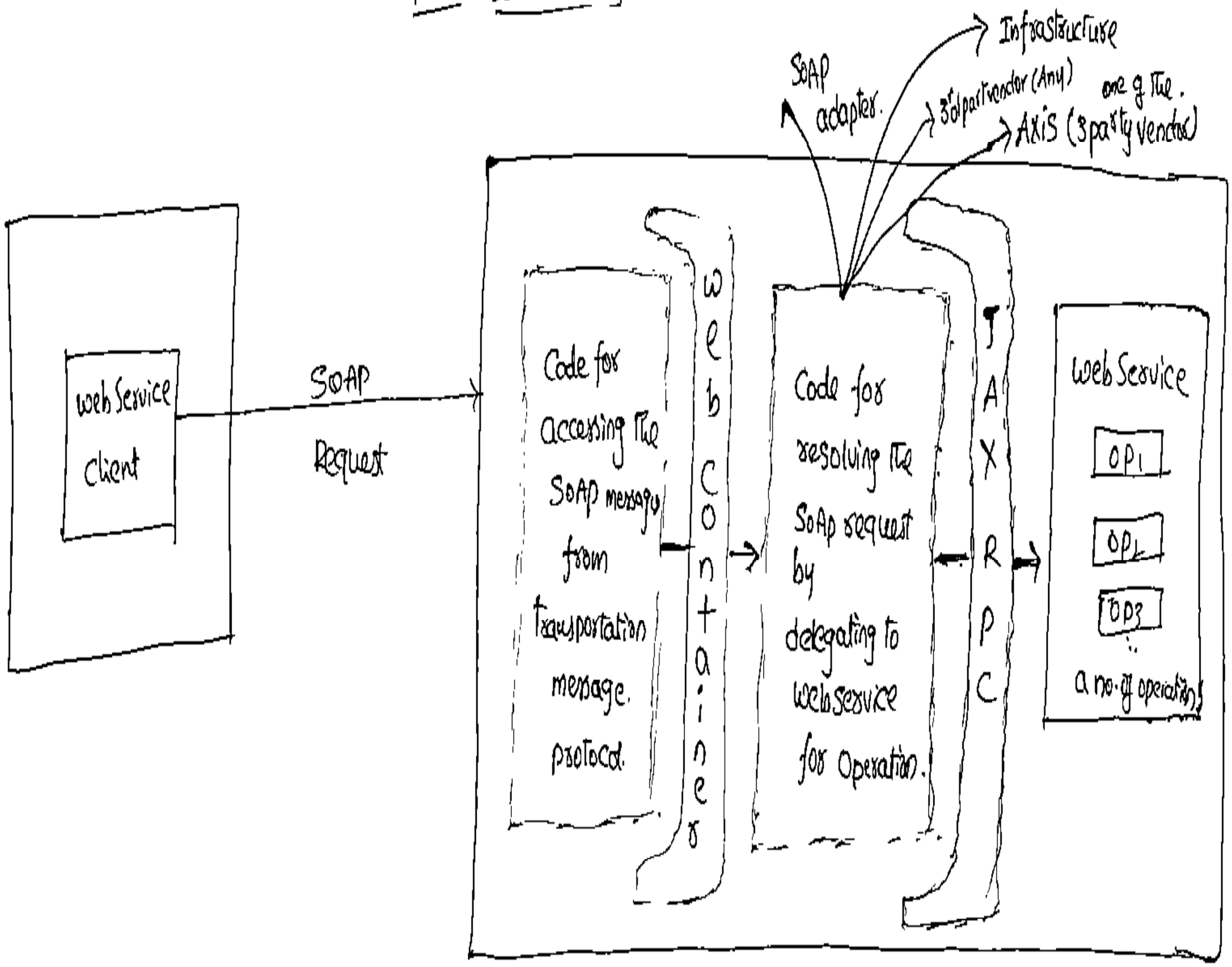
1) creating jws
2) creating wsdd (webservice deployment descriptor).

=) Now download Axis from apache website.

extract it and deploy into to java web application server. /webapps eg: Tomcat/webapps

[continue]

AWS - Server Client



=) Now a write a Simple java Object and expose it as a web Service.

```
Public class BusinessObjectTest
  {
    psvm
  Public String getMessage (String s)
  {
    S.O.pln ("in getMessage" +s);
    return "welcome to Axis from" + s;
  }

public int add (int i, int j)
  {
    S.O.pln ("in add between " + i + " : " + j);
    return (i+j);
  }

} // BusinessObjectTest
```

Compiled by Axis.
Source file

=) Save the above file as BusinessObjectTest.jws

=) Now Copy the BusinessObjectTest.jws and deployed it into Axis Webroot.

=) Then Start the Tomcat Server

=) And make a request

http://localhost:7000/axis/Doc/BusinessObjectTest.jws.

=) Click to see the WSDL, where the WSDL file describes the information like

=) Type (post)

=) message.

⇒ PostType (operation)

⇒ Binding (describes the transport and encoding)

⇒ Service.

⇒ Use the following URL to Test

http://localhost:7000/axis/DOC/BusinessObjectTest.jws?

method = getMessage & arg = test

Note:- we have various demerits in creating jws like

a) java Source is required.

b) only POJO's accepted

c) Complex Types are not accepted.

⇒ we can overcome the above demerits using WSDD

where WSDD is an xml document describing abt the Service to the Axis platform.

```
<deployment - - - - >
<Service name = "BusinessObjectTestService">
<parameter name = "javaClass"  value = " " />
<parameter name = "allowedMethods"  value = "*" />
</service>
</deployment>
```

Client side Axis Architecture

understanding accessing webservice using Axis Impl using JAX-RPC Runtime.

So Axis implementation for JAX-RPC runtime in the client Support infrastructures to create, receive SOAP message & transport.

Axis implementation provides 3 styles of accessing webservices.

a) Generated Stub.
b) Dynamic stub.
c) Dynamic Invocation Interface.

Generated Stub :

In this approach we use Vendor specific tool to create the stub of accessing webservice.

Command is

java org.apache.axis.wsdl.wsdl2java -p com.nit.ws

http://localhost:7070/axis/BusinessObjectTest.jws.

(tree(f))

```
com
  nit
    ws
      BusinessClientTestService.java
      BusinessObjectTestServiceLocator.java
      BusinessObject_PortType.java
      BusinessObjectTestSoapBindingStub.java
```

Set class for web services. we use below class jars.

axis.jar	Saaj.jar
axis-ant.jar	log4j.jar
Common-discovery.jar.	wsdl4j.jar.
Common_logging.jar.	
jax-rpc.jar.	webservices.jar.com.nit.ws.client.

```
// Client application

public class client
{
  p s v m (String ... s) throws Exception
  {
  // getting the BusinessObjectTestService.
  BusinessObjectTestService bo = new BusinessObjectTestServiceLoca
  -tor();
  BusinessObjectTest_PortType bp = bo.getBusinessObjectTest();
  System.out.println(bp.getMessage("AxisClient"));
  System.out.println(bp.add(1,1));
  }
}
```

The above approach is vendor specific, and it is difficult to migrate the application from one implementation to another, without modifying the code.

To support the above requirement, we have other styles supported by AXIS are

Dynamic stub :-

To use this style we have to write an Interface which should be a subtype of java.rmi.Remote, and this interface describes the operations supported by the webservice.

--- Steps to use the above approach

a) Get the ServiceFactory
b) Create a Service
c) Get the port, and take it to the Interface type
d) perform the Operation.

BusinessInterface.java

```
public interface BusinessInterface extends
        java.rmi.Remote {

        String getMessage(String s)
        int add(int i, int j);
        }
```

```
import javax.xml.spc.*;
import javax.xml.namespace.*;
import java.net.*;

    public class Client1
    {
        psvm(String... s1) throws Exception
    }
ServiceFactory sf = ServiceFactory.newInstance();
```

```
Service s = sf.createService(new URL("http://localhost:7000/axisDoc/BusinessObjectTest.jws?wsdl"), new
QName("http://localhost:7000/axisDoc/BusinessObjectTest.jws", "BusinessObjectTestService"));

BusinessInterface bi = (BusinessInterface)s.getPort(BusinessInterface.class);
        S.O.pln(bi.getMessage("Dynamic Client"));
            S.O.pln(bi.add(1,2));
        }
    }
```

9/10/12

In the above Approach can easily migrate from one java-rpc implementation to another. But here we have to write an I/f describing the webservices properties.

=) Apart the above approach we have another approach.

Dynamic Invocation Interface.

=) In this approach we can dynamically locate the webservice and access, and here we no need to write an interface, what we have done for the above approach

we follow the following steps to use this approach

a) Create a Service implemented by a Vendor.

b) Create a Call

c) set the operation name.

d) set the Target End Point Address

e) Invoke. the service.

client program

```
import javax. xml. namespace.*;
        "      xml.rpc.*;

    public class TestDii
    {
        psvM (String ...s) throws Exception.
    }
Service factory sf =
org. apache. axis. client. Service s1 = new org. apache. axis.
client. Service ();

    Call c = (Call)s1. createCall (new QName ("http://local
host :7000 /axisDec / BusinessObjectTest.jw", "BusinessObjectTest
Service"))

        c. setOperationName (new QName("http://localhost :
7000 /axisDec/BusinessObject Test.jws", "getMessage"));

        c. setTargetEndPointAddress("http://localhost :7000/axis1Dec
/BusinessObject Test.jws");

        Sop.println (c. invoke (new Object[ ] {"dii"}));
```

```
c.SetOperationName(new QName("urn://localhost:70
/axisDoc/BusinessObjectTest.jws", "add"));
s.println(c.invoke(new Object[] {1, 3}));
}
}
```

=) Now recall the second approach

first approach
jws → 3 files

Creating

WSDD (WebServiceDeploymentDescriptor)

=) WSDD is a xml document to describe webservice in the axis platform.

DeptBusinessObject

```
import java.sql.*

public class DeptBusinessObject
{
public boolean saveDept(int deptno, String dname,
String loc)
{
Connection con = null;
try {
Class.forName("oracle.jdbc.driver.OracleDriver");
con = DriverManager.getConnection("jdbc:oracle:thin:
@localhost:1521
```

```
: oscl", "System", "manager 123");

Statement st = con.CreateStatement();

int i = st.execuleUpdate("insert into

dept values ("+deptno+", '"+dname+"',

'"+loc+"');");

return i=10;

} //try.

Catch (Exceptione)

{

e.printStackTrace();

}

finally {

try {

con.close()

}

Catch (Exception e)

{

e.printStackTrace();

}

} //try.

return false;

}
}
```

2) Compile & deploy in Tomcat Server

=) Now write a webservice deployment descriptor

```
<deployment
  xmlns = "http://xml.apache.org/axis/wsdd/"
  xmlns:java = "http://xml.apache.org/axis/wsdd/providers
    /java">

  <service name = "DeptBusinessObjectService"
      provider = "java:RPC">
    <parameter name = "className"
      value = "DeptBusinessObject" />
  <parameter name = "allowedMethods" value = "*" />
    </service>
  </deployment>
```

=) run Tomcat

=) set classpath

=) java.org.apache.axis.client.AdminClient -p 7000
   deptbo.wsdd

=) java *.java

=) open browser type url for getting WSDL like
   http://

=) make sql command comments on the cmd prompt.

=) To deploy the webservice, use the following command, prior to this the jar files should be set into the classpath, which are in to the axis lib folder.

=) Compile the java file and deploy it into

D:\Tomcat 5.0\webapps\axis\web-INF\Classes (here)

=) into the lib folder even paste ojdbc14.jar.

=) java org.apache.axis.client.AdminClient -p 7000 deptbo.wsdd

After this to test.

In the browser.

http://localhost:7000/axis/services/DeptBusinessObjectService

wsdl file we will get.

=) Now writing client application using the generated stub approach

=) Command is.

java org.apache.axis.wsdl.WSDL2Java. -p com.bit.ws

http://localhost:7000/axis/services/DeptBusinessObjectService?wsdl

=) java file will be generated.

=) Compile them.

=) and write client location.

Client Test.java

```
pk com.nit.ws.

public class ClientTest
{
    psvm (String ... s) throw Exception.

{
    DeptBusinessObjectService_Service ds = new
    DeptBusinessObjectService_ServiceLocator();

    DeptBusinessObject dbo
    = ds.getDeptBusinessObjectService();

    S.o.p.ln (dbo.saveDept(50, "d50", "d51"));

}
}
```

10/10/12

In the previous application we have used simple types & the requirement is to use complex types, we do the following modifications.

=) write a Dept class

```
public class Dept implements Serializable
{
    int deptno;
    String dname, loc;
}
```

=) In to the Business object.

```
public class DeptBusinessObject
{
public boolean saveDept (Dept d)
{
    int deptno = d.deptno;
    String dname = d.dname;
    String loc = d.loc;
    // Remaing code is same.
```

=) In deptbo.wsdd

```
<beanmapping qname = "nitrs.Dept"
xmlns:nitrs = "urn:DeptMapping"
languageSpecificType = "java.Dept" />
</service>
</deployment>
```

=) java org.apache.axis.client.AdminClient -p 7000 deptbo.wsdd

=) http://localhost:7000/axis/services/DeptBusinessObjectService?wsdl

=) java.org.apache.axis.wsdl.WSDL2Java -p com.nit.ws

=) http://localhost:7000/axis/services/DeptBusinessObjectService?wsdl

=) // in the client application.

s.o.println (dbo. saveDept (new Dept (40, "d40", "d41"

JAX - WS

java api for xml webservices.

=) JAX-WS provides a simplified approach with some standard annotations, that simplifies, webservice application development and client in java.

=) The current version of JAX-WS is JAX-WS 2.0

=) This is much simpler to JAX-RPC, which have already seen.

=) The benifits we have into JAX-WS are

a) platform Independent Stub.

(Dynamic proxy mechanism for)

=) JAX-WS uses formal delegation model with pluggable provider support. when compared with JAX-RPC, In JAX-RPC the stub is vendor specific

b) Support for annotations

=) The support for annotations simplifies the webservices application development, where we can define the web service easily using annotations like

@ webservice.

c) Suppos for Asynchronous access.

JAX-WS Supports Asynchronously accessing webservices. It Supports Polling and call backs.

d) Supports Resource Injection.

⇒ JAX-WS Supports Resource Injection. It uses annotations given into JAX-250 for resource Injection

⇒ To use the Resource Injection we use the following annotations i.e

@ Resource

for eg:- @ webservice.

```
public class Myclass implement MyIntf
{
    @ Resource.
      WebServiceContext ws;
      public int m1(MyObject m)
        { ..... }
}
```

e) Supports JAXB and SAAJ.

⇒ JAX-WS Support multiple technologies. for data binding i.e Converting java objects into Ym document where JAXB (Continue) Supports SAAJ for data binding

f) Supports static and dynamic clients

⇒ JAX-WS supports static as well as dynamic clients. Static clients are those client that use service end point interface for accessing the web services, and the dynamic clients uses message oriented approach for accessing the web services.

g) Supports Soap 1.2

JAX-WS supports SOAP 1.1 and SOAP 1.2. SOAP 1.2 supports attachments i.e binary content like Image where it uses message optimized

h) Supports development tools :

⇒ JAX-WS supports development tools like WS gen. and WS import for ~~creating and accessing the webservice~~ generating the JAX-WS runtime elements. we have two approaches is bottom up approach and top down approach.

⇒ In bottom up approach from the java classes or components that are exposed as webservice are given as input to the tool for generating WSDL and the run-time elements

⇒ In the top down approach we create WSDL and give that as input to the tool, where the tool generates the java classes to which we add additional code and expose them as webservices.