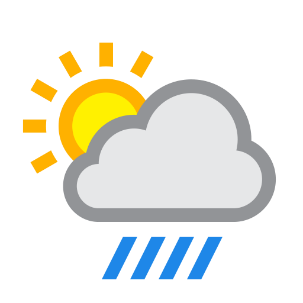
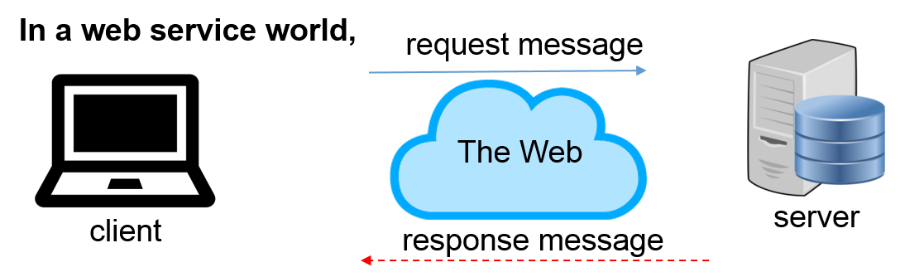
**C302 Web Services: P01 Worksheet**

Weather Forecast



# Introduction to XML

**Refer to slides 1 to 9 in “Resources\_about\_XML.pdf” for this section.**



1. XML stands for e**X**tensible **M**arkup **L**anguage. Like HTML, it is a markup language. XML, however, is designed to store and transport data.

Figure 1-1 shows a sample XML document that describes this course, C302. The XML is quite self-descriptive and readable. XML stores data in plain text format, with “.xml“ file extension.

|  |
| --- |
| <module>    <code>C302</code>    <title>Web Services</title>  <classes>  <class>1-W64A</class>  <class>2-W64B</class>  <class>3-W64C</class>  </classes>  </module> |

Figure 1-1 : module.xml

1. What is a markup language?

|  |
| --- |
| A markup language is a system for annotating a document in a way that is syntactically distinguishable from text, meaning when the document is processed for display, the markup language is not shown, and only used to format the text. |

1. In what way is XML similar to HTML?

|  |
| --- |
| XML and HTML both uses tags. |

1. Identify the differences between XML and HTML by completing Table 1-2.

|  |  |
| --- | --- |
| **Description** | **XML / HTML?** |
| Designed to display data | HTML |
| Designed to carry data | XML |
| The tags are predefined | HTML |
| The tags are “invented” by the author | XML |

Table 1-2 : XML vs HTML

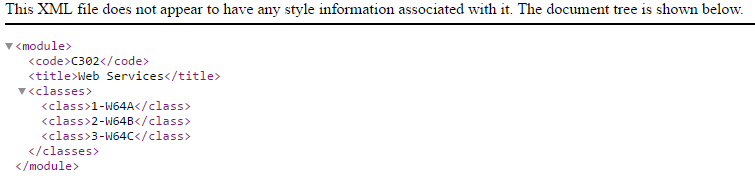
1. What other differences between XML and HTML can you find?

|  |
| --- |
| XML tags are extensible whereas HTML has limited tags |

1. How is XML extensible? (Hint: Can information be added?)

|  |
| --- |
| It is extensible because it is not fixed format like HTML. XML let you design your own markup language for limitless different type of documents. |

1. Open a text editor eg. Notepad and copy & paste the XML content in Figure 1-1 into a new file. Save the file as module.xml.
2. Open the file using a web browser eg. Chrome to view the XML document. You should see the following output:



1. Edit module.xml by removing the last line </module>. Try to view the file in a web browser. Explain briefly what you see in the browser.

|  |
| --- |
| Graphical user interface, text, application  Description automatically generated |

1. Notice that the web browser renders the content in a hierarchical structure, called a tree.

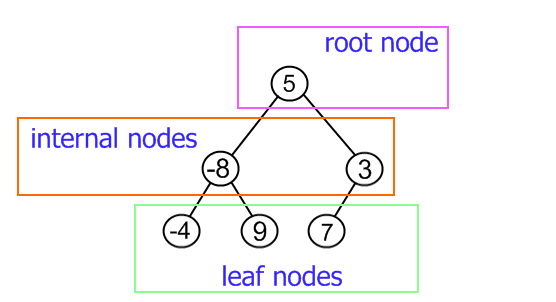
Figure 2-1 : tree structure for module.xml

A tree is a data structure that consists of:-

**Root node**: the topmost node in a tree. It is a kind of "main node" in the tree, because all other nodes can be reached from root. Also, it has no parent.

**Internal nodes**: Non-root and non-leaf nodes, these nodes have a parent (**the** **root node is not an internal node**), and at least one child.

**Leaf nodes:** these nodes has a parent, but has no children.

****

1. Look up the following article <https://en.wikipedia.org/wiki/Tree_(data_structure)> to know more about the various terminologies:-

* Root
* Parent / Ancestor
* Child / Descendant
* Sibling
* Leaf
* Internal node

1. Based on the tree structure shown in Figure 4-1,

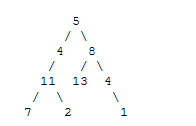


Figure 4-1 : tree structure

1. Which number represents the root node?

|  |
| --- |
| 5 |

1. Which node(s) is/are the sibling(s) of the number “13”?

|  |
| --- |
| 4 |

1. Which number(s) are the leaf nodes?

|  |
| --- |
| 7, 2, 13, 1 |

1. How many leaf element(s) are in the tree?

|  |
| --- |
| 4 |

1. Name the internal nodes.

|  |
| --- |
| 11, 4, 8, 4 |

1. Identify the children node(s) for the element 8.

|  |
| --- |
| 13, 4 |

1. Which node is the ancestor of the number “11”?

|  |
| --- |
| 4 |

1. Which node(s) is/are the descendant(s) of the number “11”?

|  |
| --- |
| 7, 2 |

## XML Tree

1. An XML document is a single Tree that allows only one root element.

|  |  |
| --- | --- |
| C:\Users\denise_quek\Desktop\Picture5.png | C:\Users\denise_quek\Desktop\Picture6.png |

Study the data structure of the tree in Figure 5-1 and write down the XML document. You may use your own data for the IDs, first name and last name.

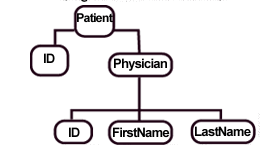
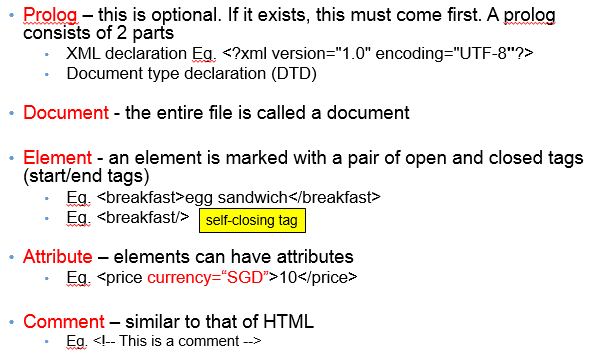


Figure 5-1

|  |
| --- |
| <patient>  <id>19013345</id>  <physician>  <id>19013345</id>  <firstname>long</firstname>  <lastname>ju</lastname>  </physician>  </patient> |

## XML Terminology, Syntax and Well-Formed Documents

1. These are the terminologies used in an XML document.



1. Syntax rules for XML documents are listed below. XML documents that conform to the syntax rules are said to be “well-formed” XML documents.

|  |
| --- |
| * XML files should be saved as Unicode UTF-8 or UTF-16 * An XML declaration is optional. If included, it must be placed as the first line in the XML document * XML documents must contain 1 root element * All XML elements must have open and closed tags   + XML element names can be alphanumeric. The only symbols allowed are hyphen (-), underscore (\_) and period (.) * XML elements must be properly nested   + <date> <day> … </day> … </date> | valid   + <date> <day> … </date>… </day> | invalid * XML tags are case-sensitive * XML attribute values must be quoted with " " or ' ' * Predefined Character Entities     Eg. <message>age &lt; 18</message> |

1. Figure 8-1 shows a XML document that stores employee information for a company.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <Company>  <Employee type="full-time">  <ID>19821732</ID>  <FirstName>Alicia</FirstName>  <LastName>Keys</LastName>  <ContactInfo>  <OfficeNo>61234567</OfficeNo>  <Email>alicia.keys@xyz.com</Email>  </ContactInfo>  </Employee>  </Company> |

Figure 8-1 : company.xml

1. Draw the tree structure (of element nodes) that represents the given XML document. Work with your team mates on this. Post your diagram in your respective MS Teams team channel

|  |
| --- |
| Timeline  Description automatically generated |

1. What is the name of the root element?

|  |
| --- |
| Company |

1. How many child elements does “ContactInfo” element have?

|  |
| --- |
| 2 |

1. What is the attribute name and value for element “Employee”?

|  |
| --- |
| ID: 19821732  FirstName: Alicia  LastName: Keys  ContactInfo: {  OfficeNo: 61234567  Email: alicia.keys@xyz.com  } |
|  |

1. Is there any empty tag in the XML document? If yes, provide the tag name.

|  |
| --- |
| no |

1. Figure 9-1 shows the breakfast menu for RP cafe. However, the XML document is NOT well-formed. Identify the issues with the XML document.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8**"**?>  <breakfastMenu>    <type=local>  <item>Nasi Lemak</item>  <item>Mee Hoon</item>  <item>Carrot Cake</item>  </type>  </breakfast\_menu> |

Figure 9-1 : breakfast.xml

Identify the issues with the XML document.

|  |
| --- |
| Breakfast menu tag isn’t the same.  The type isn’t typed correctly |

# XML DOM

**Refer to slides 10 to 12 in “Resources\_about\_XML.pdf” for this section.**

The DOM is an official recommendation of the World Wide Web Consortium (W3C). DOM is used for accessing and manipulating documents like HTML and XML. The official website is at <https://www.w3.org/DOM/>

1. What is the Document Object Model (DOM) ?

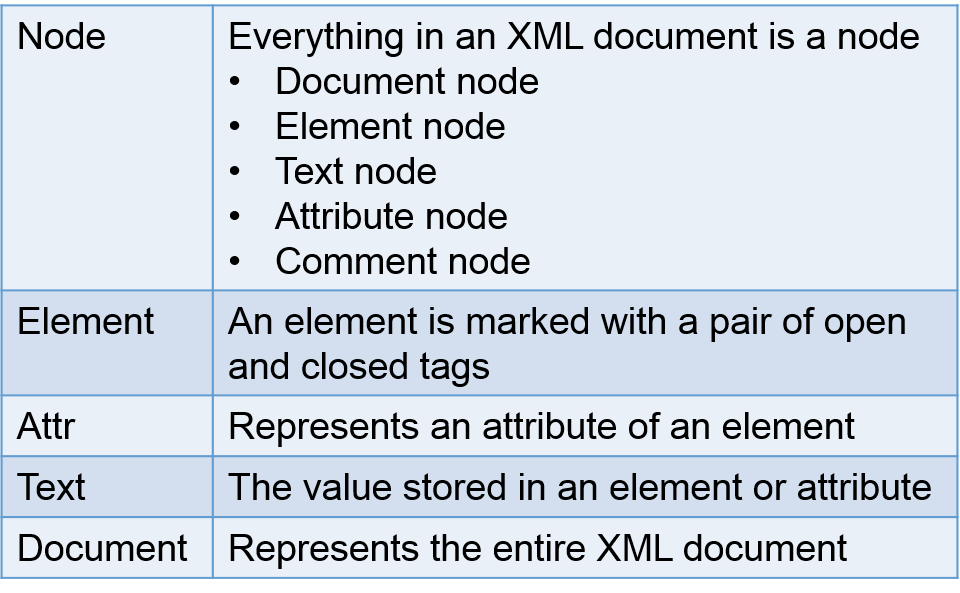
|  |
| --- |
| The Document Object Model is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure wherein each node is an object representing a part of the document. The DOM represents a document with a logical tree |

1. XML DOM is used for working with XML documents. It presents an XML document as a tree structure. All elements can be accessed through the XML DOM.

Everything in an XML document is a **node**

* 1. The entire document is a document node
  2. Every XML element is an element node
  3. The text in the XML elements are text nodes
  4. Every attribute is an attribute node
  5. Comments are comment nodes

DOM Interfaces



1. Figure 12-1 shows another sample XML document that describes this course C302.

|  |
| --- |
| <module code="C302">  <title>Web Services</title>  <year>3</year>  </module> |

Figure 12-1 : module\_c302.xml

The XML DOM structure can be represented as follows:-

|  |
| --- |
|  |

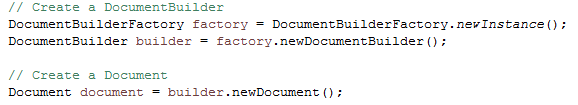
# XML Parsing using Java

**Refer to slides 13 to 16 in “Resources\_about\_XML.pdf” for this section.**

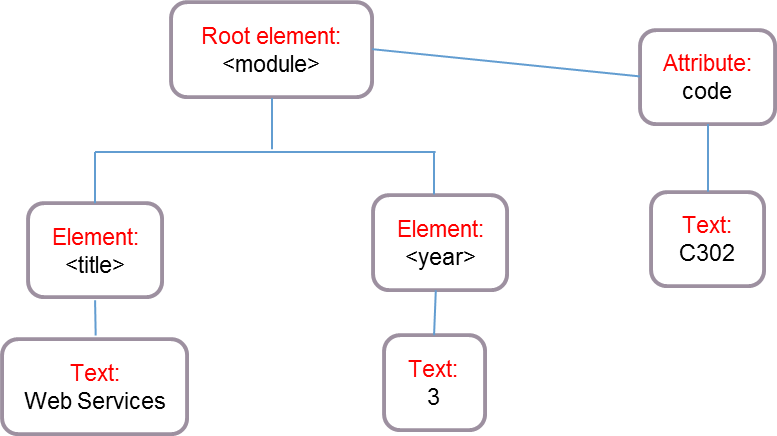
XML Parser enables us to work with XML documents. Every programming language has its own support for XML parsing. Java provides multiple options to parse XML document. We will be using the DOM parser which parses the document by loading the complete contents of the document and creating its complete hierarchical tree in memory.

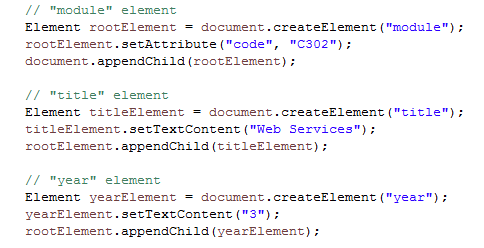
## General steps in Creating a XML file

* Firstly, create a Document with DocumentBuilder class.



* Create the XML content with Element class.
  + Create the Element object
  + Set attributes and/or text where applicable
  + Attach the element to its parent

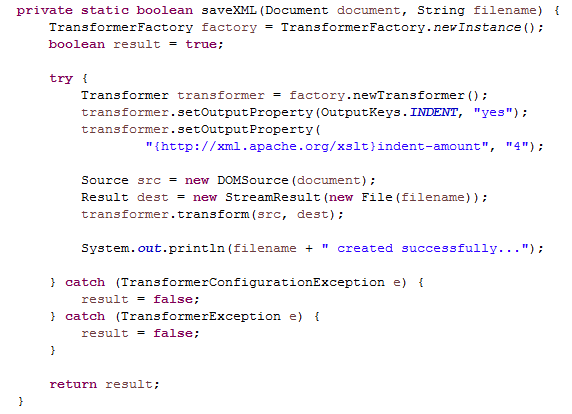




* Finally, save the XML document by using the helper method:

*Eg. saveXML(document, "module\_c302.xml");*

The helper method is defined as follows:-



## Create module\_c302.xml in Java

1. Create a new Java project in Eclipse with the following information:-

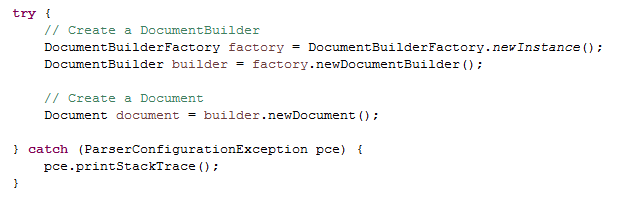
|  |  |
| --- | --- |
| Project Name: | P01 – XML |
| Java Class: | ModuleXMLBuilder.java |
| Details: | A Java program to create module.xml |

We are going to create a Java program that will create module\_c302.xml as shown below.

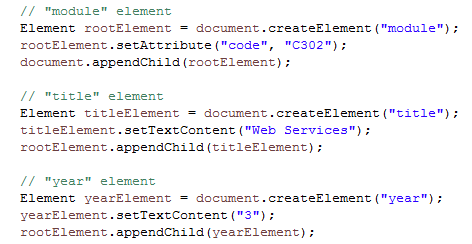
|  |
| --- |
| <module code="C302">  <title>Web Services</title>  <year>3</year>  </module> |

Figure 13-1 : module\_c302.xml

1. In the main method, create a document using DocumentBuilder.



1. After creating a document, proceed to create all the elements by adding on the following code.



1. What is the method that is used to create an element?

|  |
| --- |
| document.createElement(); |

1. What is the purpose of appendChild method?

Hint: In Eclipse, mouse over the method name. A pop-up explanation of the method will be given.

|  |
| --- |
| To add into the xml as an attribute |

1. How do we set attribute for an element?

|  |
| --- |
| setAttribute |

1. How do we set Text for an element? Is it possible to pass in an integer?

|  |
| --- |
| setTextContent. It is not possible to pass an integer |

1. Save the XML document to file by using the helper method:

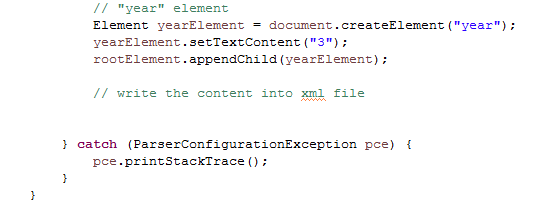
|  |
| --- |
| private static boolean saveXML(Document document, String filename) {  TransformerFactory factory = TransformerFactory.newInstance();  boolean result = true;    try {  Transformer transformer = factory.newTransformer();  transformer.setOutputProperty(OutputKeys.INDENT, "yes");  transformer.setOutputProperty(  "{http://xml.apache.org/xslt}indent-amount", "4");    Source src = new DOMSource(document);  Result dest = new StreamResult(new File(filename));  transformer.transform(src, dest);    System.out.println(filename + " created successfully...");    } catch (TransformerConfigurationException e) {  result = false;  } catch (TransformerException e) {  result = false;  }    return result;  } |

1. How do we write an XML document to file? Provide the Java API for doing so.

Hint: Look up the official API website at <https://docs.oracle.com/javase/8/docs/api/javax/xml/transform/Transformer.html>

|  |
| --- |
| javax.xml.transform.Transformer |

1. Call the method saveXML from within the main method.



Write the line of code to call saveXML().

|  |
| --- |
| *saveXML*(document, "P01.xml"); |

1. Run the program. Where is the XML file being created?

Hint: Refresh Eclipse IDE by pressing F5.

|  |
| --- |
|  |

1. Try removing the setting of output property for the transformer. Compare and contrast the differences in the output of module.xml and write down your observations.

|  |
| --- |
| Without the property of the transformer the xml will not be formatted. It will cluster all into 1 line. |

## Using Plain Old Java Object (POJO)

Since Java is object-oriented, it is common to represent things as objects. Look at the following XML. We can represent "student" as a Plain Old Java Object (POJO).

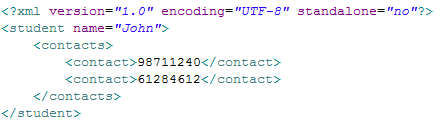
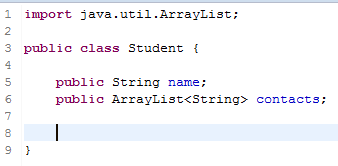


Figure 17-1: student.xml

1. What are the attributes/fields that a student should have? Indicate the data type as well.

|  |
| --- |
| Email, password, studentID. The data type should be TextContext |

1. Create Student.java as follows:-



1. Generate the constructor, getter and setter methods for each attribute/field.

Hint: Right-click 🡪 Source 🡪 Generate Constructors using Fields…

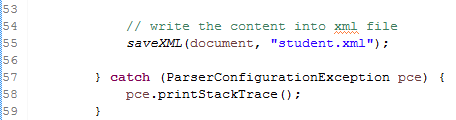
🡪 Generate Getters and Setters…

Student.java

|  |
| --- |
| *public class* Student {    String name;  ArrayList<String> contacts;   *public* Student(String name, ArrayList<String> contacts) {  *this*.name = name;  *this*.contacts = contacts;  }   *public* String getName() {  *return* name;  }   *public void* setName(String name) {  *this*.name = name;  }   *public* ArrayList<String> getContacts() {  *return* contacts;  }   *public void* setContacts(ArrayList<String> contacts) {  *this*.contacts = contacts;  } } |

1. Create the main program StudentXMLBuilder.java. In the main method, include the following:-





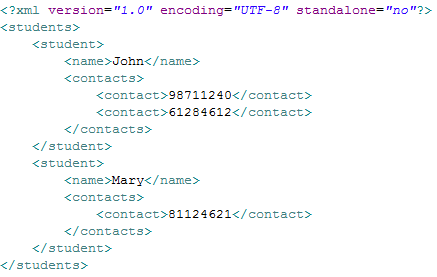
1. Which lines of codes are used to create a student?

|  |
| --- |
| *Element* rootElement = document.createElement("student"); rootElement.setAttribute("name", student.getName()); document.appendChild(rootElement); |

1. Explain the purpose of the for loop at line 45.

|  |
| --- |
| To get each contact within the arraylist contacts and set into a each individual tag. |

1. Include the saveXML method and run the program. The file student.xml will be generated.
2. By using Plain Old Java Object, write a program StudentsXMLBuilder.java to create students.xml as shown below.



# Solving the Problem

1. Task 1a: Write a Java program ***DailyForecastApp.java*** to write the following XML document to a file.

Task 1b: Enhance your program by using POJO to represent a Daily Forecast object. Write the program in ***DailyForecastApp\_pojo.java***.

The content of the XML document is as follows:

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

<dailyForecast>

<date>2018-04-01</date>

<description>Sunny</description>

<temperatures>

<maxTemp unit="C">32</maxTemp>

<minTemp unit="C">27</minTemp>

</temperatures>

<windSpeed unit="kph">3</windSpeed>

</dailyForecast>

Figure 25-1 : daily\_forecast.xml

1. Task 2: Design and create XML for a three-day forecast. Name your XML file as three\_days\_forecast.xml. Note: Writing a Java program to write the XML document to a file is not required. You can validate and format your XML using an online XML validator Eg. <http://www.xmlvalidation.com/>

Data is as shown in Table 25-1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| date | Description | maxTemp | minTemp | windSpeed |
| 2018-04-01 | Sunny | 32 | 27 | 3 |
| 2018-04-02 | Rainy | 28 | 23 | 7 |
| 2018-04-03 | Cloudy | 30 | 25 | 5 |

Table 26-1 : data for three-day forecast

1. Enhancement – The following is a simplified 1 day forecast from NEA website.
2. Write a Java program ***NEA\_WeatherApp.java*** to create the following XML.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" standalone="no"?>  <feed xmlns="http://www.w3.org/2005/Atom">  <title>1 Day Forecast</title>  <link href="http://www.nea.gov.sg/weather\_3day.aspx/" rel="self"/>  <link href="http://www.nea.gov.sg/"/>  <updated>2018-03-31T10:13:00Z</updated>  <author>  <name>Meteorological Service Singapore</name>  </author>  <id>urn:uuid:20180331-d399-11d9-b93C-051405003af6</id>  <content type="xhtml">  <div xmlns="http://www.w3.org/1999/xhtml">  <img src=" http://www.weather.gov.sg/wp-content/themes/wiptheme/assets/img/icon-fair-warm.png"/>  <forecast>Fair and warm.</forecast>  </div>  </content>  </feed> |

1. Open the XML using a web browser. Write your observations here.

|  |
| --- |
| Graphical user interface, application  Description automatically generated |

**End of worksheet**