1. Do the workshop in the chapter 9
2. After finishing the workshop students must complete the following tasks:

XML DOM Objects:

- Adding nodes
- Removing nodes
- Replacing nodes

SAX:

- Parsing XML document
- Finding Elements by name
- Modifying XML document

3. XML DOM

In this lab, students will use the **Book.xml** and **loadxmldoc.js** files provided below:

Book.xml file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Edited by XMLSpy® -->
<bookstore>
<book category="cooking">
<title lang="en">Everyday Italian</title>
<author>Giada De Laurentiis</author>
<year>2005</year>
<price>30.00</price>
</book>
<book category="children">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
<book category="web">
<title lang="en">XQuery Kick Start</title>
<author>James McGovern</author>
<author>Per Bothner</author>
<author>Kurt Cagle</author>
<author>James Linn</author>
<author>Vaidyanathan Nagarajan</author>
<year>2003</year>
<price>49.99</price>
</book>
<book category="web" cover="paperback">
<title lang="en">Learning XML</title>
<author>Erik T. Ray</author>
<year>2003</year>
<price>39.95</price>
</book>
</bookstore>
```

loadxmldoc.js file:

```
function loadXMLDoc(dname)
{
try //Internet Explorer
  {
  xmlDoc=new ActiveXObject("Microsoft.XMLDOM");
  }
catch(e)
  {
  try //Firefox, Mozilla, Opera, etc.
    {
    xmlDoc=document.implementation.createDocument("","",null);
    }
  catch(e) {alert(e.message)}
  }
try
  {
  xmlDoc.async=false;
  xmlDoc.load(dname);
  return(xmlDoc);
  }
catch(e) {alert(e.message)}
return(null);
}
```

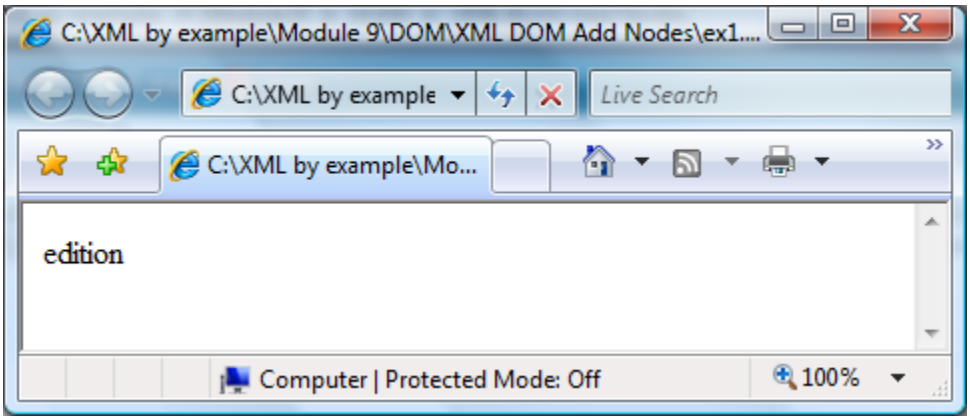
a. Adding nodes:

Let take a look at the following example: type and save as **AddingNode.html** file:

```
<html>
<head>
<script type="text/javascript" src="loadxmldoc.js">
</script>
</head>
<body>

<script type="text/javascript">
xmlDoc=loadXMLDoc("books.xml");

newel=xmlDoc.createElement("edition");

x=xmlDoc.getElementsByTagName("book")[0];
x.appendChild(newel);

document.write(x.getElementsByTagName("edition")[0].nodeName);
</script>
</body>
</html>
```

Run this file in the Internet Explorer browser

The output:

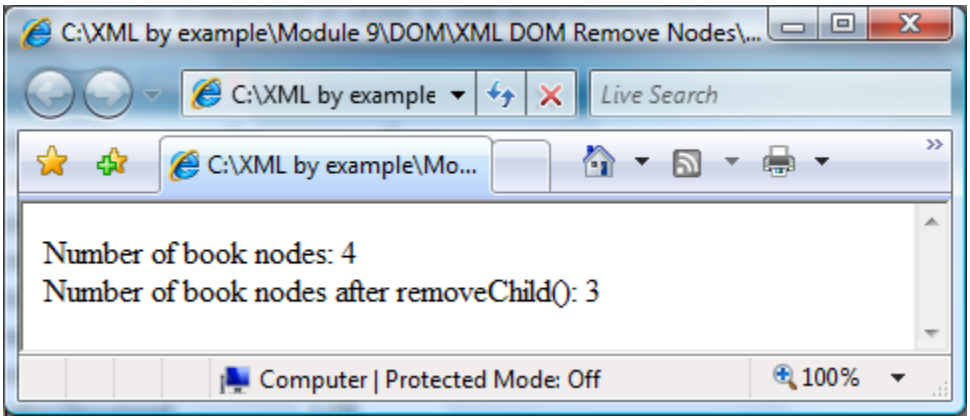


b. Removing nodes:

Type and save as **RemovingNode.html** file:

```
<html>
<head>
<script type="text/javascript" src="loadxmldoc.js">
</script>
</head>
<body>
<script type="text/javascript">
xmlDoc=loadXMLDoc("books.xml");

document.write("Number of book nodes: ");
document.write(xmlDoc.getElementsByTagName('book').length);
document.write("<br />");

y=xmlDoc.getElementsByTagName("book")[0];
xmlDoc.documentElement.removeChild(y);

document.write("Number of book nodes after removeChild(): ");
document.write(xmlDoc.getElementsByTagName('book').length);

</script>
</body>
</html>
```

Run this file in the Internet Explorer browser.
The output:



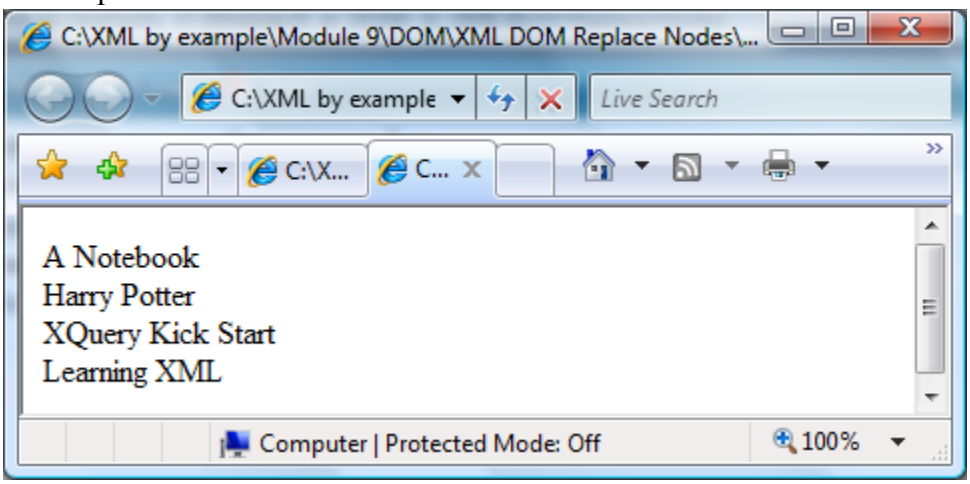
c. Replacing nodes:

Type and save as **ReplacingNode.html** file:

```
<html>
<head>
<script type="text/javascript" src="loadxmldoc.js">
</script>
</head>
<body>

<script type="text/javascript">
xmlDoc=loadXMLDoc("books.xml");

x=xmlDoc.documentElement;

//create a book element, title element and a text node
newNode=xmlDoc.createElement("book");
newTitle=xmlDoc.createElement("title");
newText=xmlDoc.createTextNode("A Notebook");

//add the text node to the title node,
newTitle.appendChild(newText);
//add the title node to the book node
newNode.appendChild(newTitle);

y=xmlDoc.getElementsByTagName("book")[0]
//replace the first book node with the new node
x.replaceChild(newNode,y);

z=xmlDoc.getElementsByTagName("title");
for (i=0;i<z.length;i++)
{
document.write(z[i].childNodes[0].nodeValue);
document.write("<br />");
}
</script>
</body>
```

```
</html>
```

Run this file in the Internet Explorer browser.
The output:



4. SAX

In this lab, examples might be using following file for testing:
Session.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<session>
   <committee type="monetary">
       <title>Finance</title>
       <number>17</number>
       <subject>Donut Costs</subject>
       <date>7/15/2005</date>
       <attendees>
           <senator status="present">
               <firstName>Thomas</firstName>
               <lastName>Smith</lastName>
           </senator>
           <senator status="absent">
               <firstName>Frank</firstName>
               <lastName>McCoy</lastName>
           </senator>
           <senator status="present">
               <firstName>Jay</firstName>
               <lastName>Jones</lastName>
           </senator>
       </attendees>
   </committee>
</session>
```

a. Parsing XML document

Type and save as **ParsingXMLDocument.java** file

```
import java.io.*;
import org.xml.sax.*;
```

```
import javax.xml.parsers.*;
import org.xml.sax.helpers.DefaultHandler;

public class ParsingXMLDocument extends DefaultHandler
{
    static int numberLines = 0;
    static String indentation = "";
    static String displayText[] = new String[1000];

    public static void main(String args[])
    {
        ParsingXMLDocument parser = new ParsingXMLDocument();
        parser.childLoop(args[0]);

        for(int loopIndex = 0; loopIndex < numberLines; loopIndex++){
            System.out.println(displayText[loopIndex]);
        }
    }

    public void childLoop(String uri)
    {
        DefaultHandler saxHandler = this;
        SAXParserFactory saxFactory = SAXParserFactory.newInstance();
        try {
            SAXParser saxParser = saxFactory.newSAXParser();
            saxParser.parse(new File(uri), saxHandler);
        } catch (Throwable t) {}
    }

    public void startDocument()
    {
        displayText[numberLines] = indentation;
        displayText[numberLines] += "<?xml version=\"1.0\" encoding=\""+
            "UTF-8" + "\"?>";
        numberLines++;
    }

    public void processingInstruction(String target, String data)
    {
        displayText[numberLines] = indentation;
        displayText[numberLines] += "<?";
        displayText[numberLines] += target;
        if (data != null && data.length() > 0) {
            displayText[numberLines] += ' ';
            displayText[numberLines] += data;
        }
        displayText[numberLines] += "?>";
        numberLines++;
    }

    public void startElement(String uri, String localName,
        String qualifiedName, Attributes attributes)
    {
        displayText[numberLines] = indentation;
```

```
        indentation += "    ";

        displayText[numberLines] += '<';
        displayText[numberLines] += qualifiedName;
        if (attributes != null) {
            int numberAttributes = attributes.getLength();
            for (int loopIndex = 0; loopIndex < numberAttributes;
loopIndex++){
                displayText[numberLines] += ' ';
                displayText[numberLines] +=
attributes.getQName(loopIndex);
                displayText[numberLines] += "=\"";
                displayText[numberLines] +=
attributes.getValue(loopIndex);
                displayText[numberLines] += '"';
            }
        }
        displayText[numberLines] += '>';
        numberLines++;
    }

    public void characters(char characters[], int start, int length)
    {
        String characterData = (new String(characters, start,
length)).trim();
        if(characterData.indexOf("\n") < 0 && characterData.length() > 0)
{
            displayText[numberLines] = indentation;
            displayText[numberLines] += characterData;
            numberLines++;
        }
    }

    public void ignorableWhitespace(char characters[], int start, int
length)
    {
        //characters(characters, start, length);
    }

    public void endElement(String uri, String localName, String
qualifiedName)
    {
        indentation = indentation.substring(0, indentation.length() - 4);
        displayText[numberLines] = indentation;
        displayText[numberLines] += "</";
        displayText[numberLines] += qualifiedName;
        displayText[numberLines] += '>';
        numberLines++;
    }

    public void warning(SAXParseException exception)
    {
        System.err.println("Warning: " +
            exception.getMessage());
    }
```

```
    public void error(SAXParseException exception)
    {
        System.err.println("Error: " +
            exception.getMessage());
    }

    public void fatalError(SAXParseException exception)
    {
        System.err.println("Fatal error: " +
            exception.getMessage());
    }
}
```

Compile and run this file for testing

```
javac ParsingXMLDocument.java
java ParsingXMLDocument "session.xml"
```

The output:

```
C:\Windows\system32\cmd.exe
C:\XML by example\Module 9\SAX>java ParsingXMLDocument "session.xml"
<?xml version="1.0" encoding="UTF-8"?>
<session>
    <committee type="monetary">
        <title>
            Finance
        </title>
        <number>
            17
        </number>
        <subject>
            Donut Costs
        </subject>
        <date>
            7/15/2005
        </date>
        <attendees>
            <senator status="present">
                <firstName>
                    Thomas
                </firstName>
                <lastName>
                    Smith
                </lastName>
            </senator>
            <senator status="absent">
                <firstName>
                    Frank
                </firstName>
                <lastName>
                    McCoy
                </lastName>
            </senator>
            <senator status="present">
                <firstName>
                    Jay
                </firstName>
                <lastName>
                    Jones
                </lastName>
            </senator>
        </attendees>
    </committee>
</session>

C:\XML by example\Module 9\SAX>pause
```

b. Finding elements by name:

Type and save as **FindingElementsByName.java** file

```
import java.io.*;
import org.xml.sax.*;
import javax.xml.parsers.*;
import org.xml.sax.helpers.DefaultHandler;

public class FindingElementsByName extends DefaultHandler
{
    static int numberLines = 0;
    static String indentation = "";
    static String displayText[] = new String[1000];

    static boolean displayBoolean;
    static String findNode;

    public static void main(String args[])
    {
        FindingElementsByName obj = new FindingElementsByName();
        findNode = args[1];
        obj.childLoop(args[0]);

        for(int index = 0; index < numberLines; index++){
            System.out.println(displayText[index]);
        }
    }

    public void childLoop(String uri)
    {
        DefaultHandler saxHandler = this;
        SAXParserFactory saxFactory = SAXParserFactory.newInstance();
        try {
            SAXParser saxParser = saxFactory.newSAXParser();
            saxParser.parse(new File(uri), saxHandler);
        } catch (Throwable t) {}
    }

    public void startDocument()
    {
        if(displayBoolean){
            displayText[numberLines] = indentation;
            displayText[numberLines] += "<?xml version=\"1.0\"
encoding=\""+
                "UTF-8" + "\"?>";
            numberLines++;
        }
    }

    public void processingInstruction(String target, String data)
    {
        if(displayBoolean){
            displayText[numberLines] = indentation;
            displayText[numberLines] += "<?";
```

```
            displayText[numberLines] += target;
            if (data != null && data.length() > 0) {
                displayText[numberLines] += ' ';
                displayText[numberLines] += data;
            }
            displayText[numberLines] += "?>";
            numberLines++;
        }
    }

    public void startElement(String uri, String localName,
        String qualifiedName, Attributes attributes)
    {
        if(qualifiedName.equals(findNode)) {
            displayBoolean=true;
        }

        if (displayBoolean){
            displayText[numberLines] = indentation;

            indentation += "    ";

            displayText[numberLines] += '<';
            displayText[numberLines] += qualifiedName;
            if (attributes != null) {
                int numberAttributes = attributes.getLength();
                for (int loopIndex = 0; loopIndex < numberAttributes;
                    loopIndex++) {
                    displayText[numberLines] += ' ';
                    displayText[numberLines] +=
attributes.getQName(loopIndex);
                    displayText[numberLines] += "=\"";
                    displayText[numberLines] +=
attributes.getValue(loopIndex);
                    displayText[numberLines] += '"';
                }
            }
            displayText[numberLines] += '>';
            numberLines++;
        }
    }

    public void characters(char characters[], int start, int length) {
        if(displayBoolean){
            String characterData =
                (new String(characters, start, length)).trim();
            if(characterData.indexOf("\n") < 0 && characterData.length()
> 0) {
                displayText[numberLines] = indentation;
                displayText[numberLines] += characterData;
                numberLines++;
            }
        }
    }
```

```
    public void ignorableWhitespace(char characters[], int start, int
length)
    {
        if(displayBoolean){
            //characters(ch, start, length);
        }
    }

    public void endElement(String uri, String localName, String
qualifiedName)
    {
        if(displayBoolean){
            indentation = indentation.substring(0, indentation.length() -
4) ;
            displayText[numberLines] = indentation;
            displayText[numberLines] += "</";
            displayText[numberLines] += qualifiedName;
            displayText[numberLines] += '>';
            numberLines++;
        }
        if(qualifiedName.equals(findNode)){
            displayBoolean=false;
        }
    }

    public void warning(SAXParseException exception)
    {
        System.err.println("Warning: " +
            exception.getMessage());
    }

    public void error(SAXParseException exception)
    {
        System.err.println("Error: " +
            exception.getMessage());
    }

    public void fatalError(SAXParseException exception)
    {
        System.err.println("Fatal error: " +
            exception.getMessage());
    }
}
```

Compile and run this file for testing

```
javac FindingElementsByName.java
java FindingElementsByName "session.xml" senator
```

The output:

```
C:\Windows\system32\cmd.exe
C:\XML by example\Module 9\SAX>java FindingElementsByName "session.xml" senator

<senator status="present">
    <firstName>
        Thomas
    </firstName>
    <lastName>
        Smith
    </lastName>
</senator>
<senator status="absent">
    <firstName>
        Frank
    </firstName>
    <lastName>
        McCoy
    </lastName>
</senator>
<senator status="present">
    <firstName>
        Jay
    </firstName>
    <lastName>
        Jones
    </lastName>
</senator>
```

c. Modifying XML document:

Type and save as **ModifyingXMLDocument.java** file

```
import java.io.*;
import org.xml.sax.*;
import javax.xml.parsers.*;
import org.xml.sax.helpers.DefaultHandler;

public class ModifyingXMLDocument extends DefaultHandler
{
    static String displayText[] = new String[1000];
    static int numberLines = 0;
    static String indentation = "";

    public static void main(String args[])
    {
        ModifyingXMLDocument obj = new ModifyingXMLDocument();
        obj.childLoop(args[0]);

        try {
            FileWriter filewriter = new FileWriter("new.xml");

            for(int loopIndex = 0; loopIndex < numberLines; loopIndex++){
                filewriter.write(displayText[loopIndex].toCharArray());
                filewriter.write('\n');
            }

            filewriter.close();
        }
        catch (Exception e) {
            e.printStackTrace(System.err);
        }
    }
```

```
    public void childLoop(String uri)
    {
        DefaultHandler handler = this;
        SAXParserFactory factory = SAXParserFactory.newInstance();
        try {
            SAXParser saxParser = factory.newSAXParser();
            saxParser.parse(new File(uri), handler);
        } catch (Throwable t) {}
    }

    public void startDocument()
    {
        displayText[numberLines] = indentation;
        displayText[numberLines] += "<?xml version=\"1.0\" encoding=\""+
            "UTF-8" + "\"?>";
        numberLines++;
    }

    public void processingInstruction(String target, String data)
    {
        displayText[numberLines] = indentation;
        displayText[numberLines] += "<?";
        displayText[numberLines] += target;
        if (data != null && data.length() > 0) {
            displayText[numberLines] += ' ';
            displayText[numberLines] += data;
        }
        displayText[numberLines] += "?>";
        numberLines++;
    }

    public void startElement(String uri, String localName,
        String qualifiedName, Attributes attributes)
    {
        displayText[numberLines] = indentation;

        indentation += "    ";

        displayText[numberLines] += '<';
        displayText[numberLines] += qualifiedName;
        if (attributes != null) {
            int numberAttributes = attributes.getLength();
            for (int loopIndex = 0; loopIndex < numberAttributes;
loopIndex++){
                displayText[numberLines] += ' ';
                displayText[numberLines] +=
attributes.getQName(loopIndex);
                displayText[numberLines] += "=\"";
                displayText[numberLines] +=
attributes.getValue(loopIndex);
                displayText[numberLines] += '"';
            }
        }
        displayText[numberLines] += '>';
```

```
        numberLines++;
    }

    public void characters(char characters[], int start, int length)
    {
        String characterData = (new String(characters, start,
length)).trim();
        if(characterData.indexOf("\n") < 0 && characterData.length() > 0)
{
            displayText[numberLines] = indentation;
            displayText[numberLines] += characterData;
            numberLines++;
        }
    }

    public void ignorableWhitespace(char characters[], int start, int
length)
    {
        //characters(characters, start, length);
    }

    public void endElement(String uri, String localName, String
qualifiedName)
    {
        indentation = indentation.substring(0, indentation.length() - 4)
;
        displayText[numberLines] = indentation;
        displayText[numberLines] += "</";
        displayText[numberLines] += qualifiedName;
        displayText[numberLines] += '>';
        numberLines++;

        if (qualifiedName.equals("lastName")) {
            startElement("", "elected", "elected", null);
            characters("2004".toCharArray(), 0, "2004".length());
            endElement("", "elected", "elected");
        }
    }

    public void warning(SAXParseException exception)
    {
        System.err.println("Warning: " +
            exception.getMessage());
    }

    public void error(SAXParseException exception)
    {
        System.err.println("Error: " +
            exception.getMessage());
    }

    public void fatalError(SAXParseException exception)
    {
        System.err.println("Fatal error: " +
            exception.getMessage());
```

```
    }
}
```

Compile and run this file for testing

```
javac ModifyingXMLDocument.java
java ModifyingXMLDocument "session.xml"
```

The output:
After you run this code, you get the result, **new.xml**, complete with the new `<elected>` elements. Check this file on your drive.

Do It Yourself

5.1. Using SAX add a new book to the below xml file with the following data:

Category: 'Computing'

Title: lang='en', XML By Example

Author: Aptech

Year: 2009

Price: 100.0

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Edited by XMLSpy® -->

<bookstore>

<book category="cooking">

<title lang="en">Everyday Italian</title>

<author>Giada De Laurentiis</author>

<year>2005</year>

<price>30.00</price>
```

```
</book>

<book category="children">

<title lang="en">Harry Potter</title>

<author>J K. Rowling</author>

<year>2005</year>

<price>29.99</price>

</book>

<book category="web">

<title lang="en">XQuery Kick Start</title>

<author>James McGovern</author>

<author>Per Bothner</author>

<author>Kurt Cagle</author>

<author>James Linn</author>

<author>Vaidyanathan Nagarajan</author>

<year>2003</year>

<price>49.99</price>

</book>

<book category="web" cover="paperback">

<title lang="en">Learning XML</title>

<author>Erik T. Ray</author>

<year>2003</year>

<price>39.95</price>

</book>

</bookstore>
```