

Lab Sheet : Introduction to XML and Basic Operations

2. Creating Your First XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<library>
```

```
  <book>
```

```
    <title>The Great Gatsby</title>
```

```
    <author>F. Scott Fitzgerald</author>
```

```
    <year>1925</year>
```

```
    <genre>Fiction</genre>
```

```
  </book>
```

```
  <book>
```

```
    <title>To Kill a Mockingbird</title>
```

```
    <author>Harper Lee</author>
```

```
    <year>1960</year>
```

```
    <genre>Fiction</genre>
```

```
  </book>
```

```
  <book>
```

```
    <title>1984</title>
```

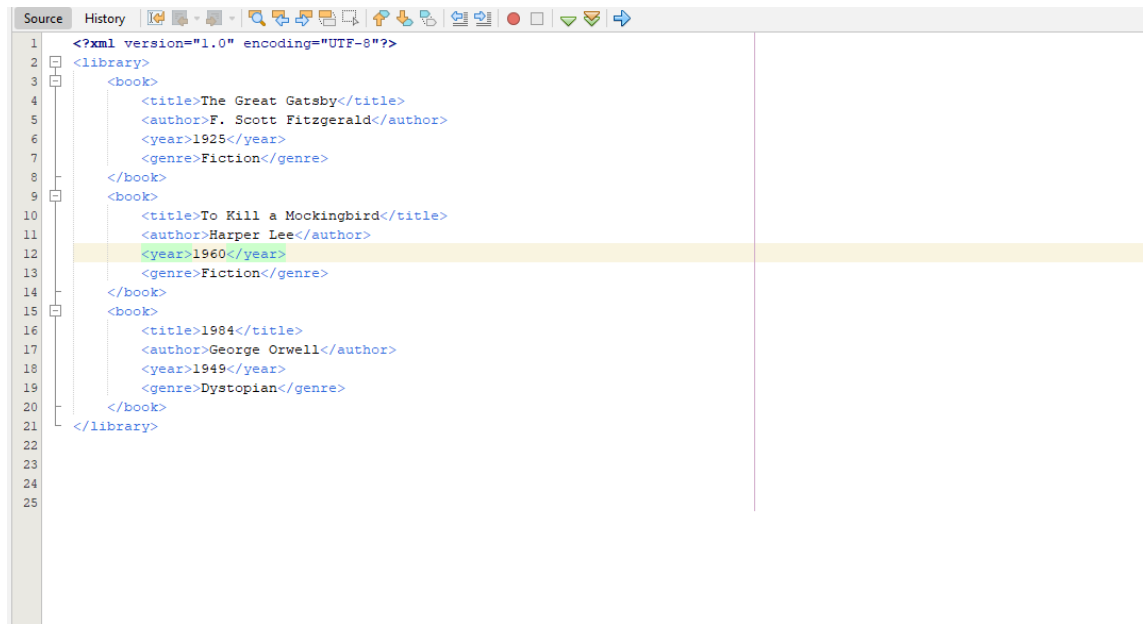
```
    <author>George Orwell</author>
```

```
    <year>1949</year>
```

```
    <genre>Dystopian</genre>
```

```
  </book>
```

```
</library>
```



3. Parsing XML in Java

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;
import java.io.File;

public class XmlParser {
    public static void main(String[] args) {
        try {
            // Create a new DocumentBuilderFactory and DocumentBuilder
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();

            // Parse the XML file
            File xmlFile = new File("books.xml");
```

```

Document document = builder.parse(xmlFile);

// Normalize the document
document.getDocumentElement().normalize();

// Get the list of <book> elements (corrected tag)
NodeList nodeList = document.getElementsByTagName("book");

// Loop through each <book> element
for (int i = 0; i < nodeList.getLength(); i++) {
    Node node = nodeList.item(i);

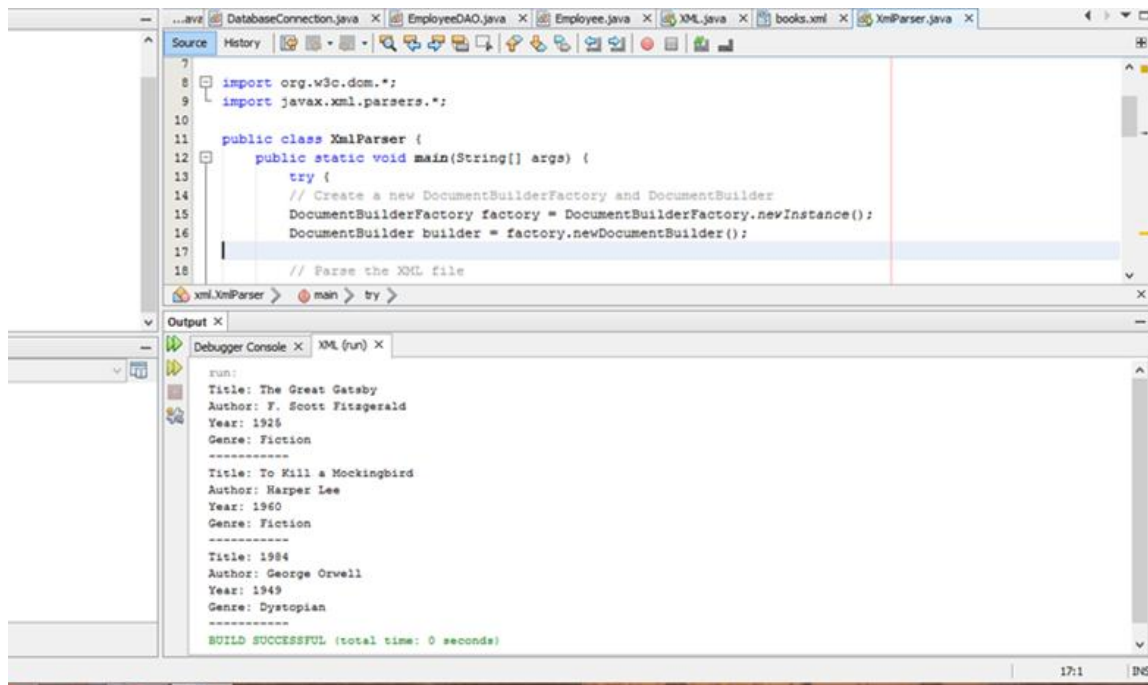
    if (node.getNodeType() == Node.ELEMENT_NODE) {
        Element element = (Element) node;

        // Get and print details of each book
        String title = element.getElementsByTagName("title").item(0).getTextContent();
        String author = element.getElementsByTagName("author").item(0).getTextContent();
        String year = element.getElementsByTagName("year").item(0).getTextContent();
        String genre = element.getElementsByTagName("genre").item(0).getTextContent();

        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Year: " + year);
        System.out.println("Genre: " + genre);
        System.out.println("-----");
    }
}

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

```



4. Modifying XML Data

// Modify the year of the first book

```
Element firstBook = (Element) nodeList.item(0);
```

```
firstBook.getElementsByTagName("year").item(0).setTextContent("2023");
```

// Save the modified document to updated_books.xml

```
TransformerFactory transformerFactory = TransformerFactory.newInstance();
```

```
Transformer transformer = transformerFactory.newTransformer();
```

```
DOMSource source = new DOMSource(document);
```

```
StreamResult result = new StreamResult(new File("updated_books.xml"));
```

```
transformer.transform(source, result); public static void main(String[] args) throws  
InterruptedException {
```

```
Counter counter=new Counter();
```

```
// Create and start multiple threads
```

```

Thread thread1 = new SynchronizedExample(counter);

Thread thread2 = new SynchronizedExample(counter);

thread1.start();

thread2.start();

// Wait for threads to finish

thread1.join();

thread2.join();

System.out.println("Final counter value: " + counter.getCount());

```

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'javaEAPracticals' with sub-packages 'Source Packages' and 'Test Packages'. The 'Source Packages' folder contains 'DatabaseConnection.java', 'ListOfBooks.xml', and 'xmlPaeser.java'.
- Source Editor:** Displays the code in 'xmlPaeser.java'. The code reads an XML file 'books.xml', prints details for each book, and updates the year of the first book from 1925 to 2023. It uses DOM4J and Transformer libraries.
- Output Console:** Shows the execution results of the program. It prints the details of three books and confirms that the XML file was updated and saved as 'updated_books.xml'.

```

// Get and print details of each book
String title = element.getElementsByTagName(name: "title").item(index: 0).getTextContent();
String author = element.getElementsByTagName(name: "author").item(index: 0).getTextContent();
String year = element.getElementsByTagName(name: "year").item(index: 0).getTextContent();
String genre = element.getElementsByTagName(name: "genre").item(index: 0).getTextContent();

System.out.println("Title: " + title);
System.out.println("Author: " + author);
System.out.println("Year: " + year);
System.out.println("Genre: " + genre);
System.out.println(x: "-----");

// Modify the year of the first book
Element firstBook = (Element) nodeList.item(index: 0);
firstBook.getElementsByTagName(name: "year").item(index: 0).setTextContent(textContent: "2023");

// Save the modified document to updated_books.xml
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
DOMSource source = new DOMSource(document);

```

Output - javaEAPracticals (run)

```

run:
Title: The Great Gatsby
Author: F. Scott Fitzgerald
Year: 1925
Genre: Fiction
-----
Title: To Kill a Mockingbird
Author: Harper Lee
Year: 1960
Genre: Fiction
-----
Title: 1984
Author: George Orwell
Year: 1949
Genre: Dystopian
-----
XML file updated and saved as updated_books.xml
BUILD SUCCESSFUL (total time: 0 seconds)

```