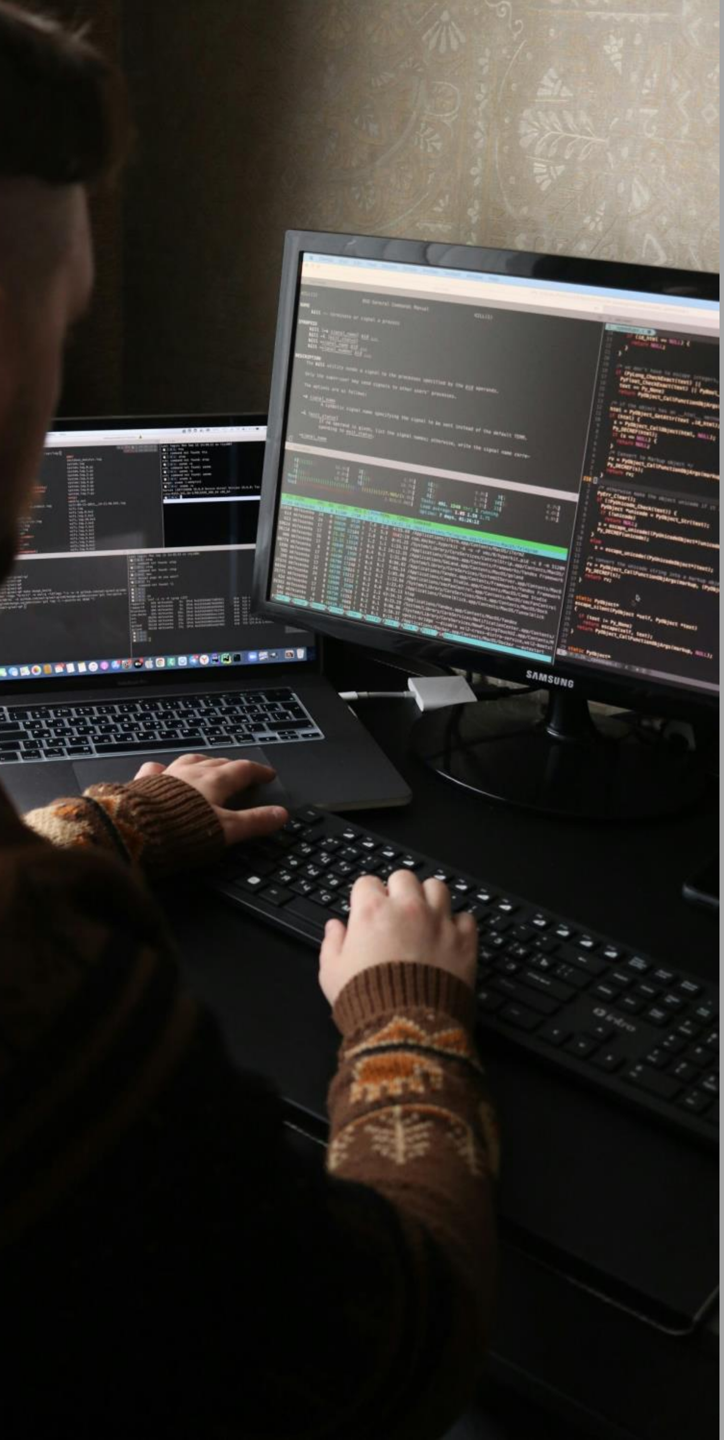
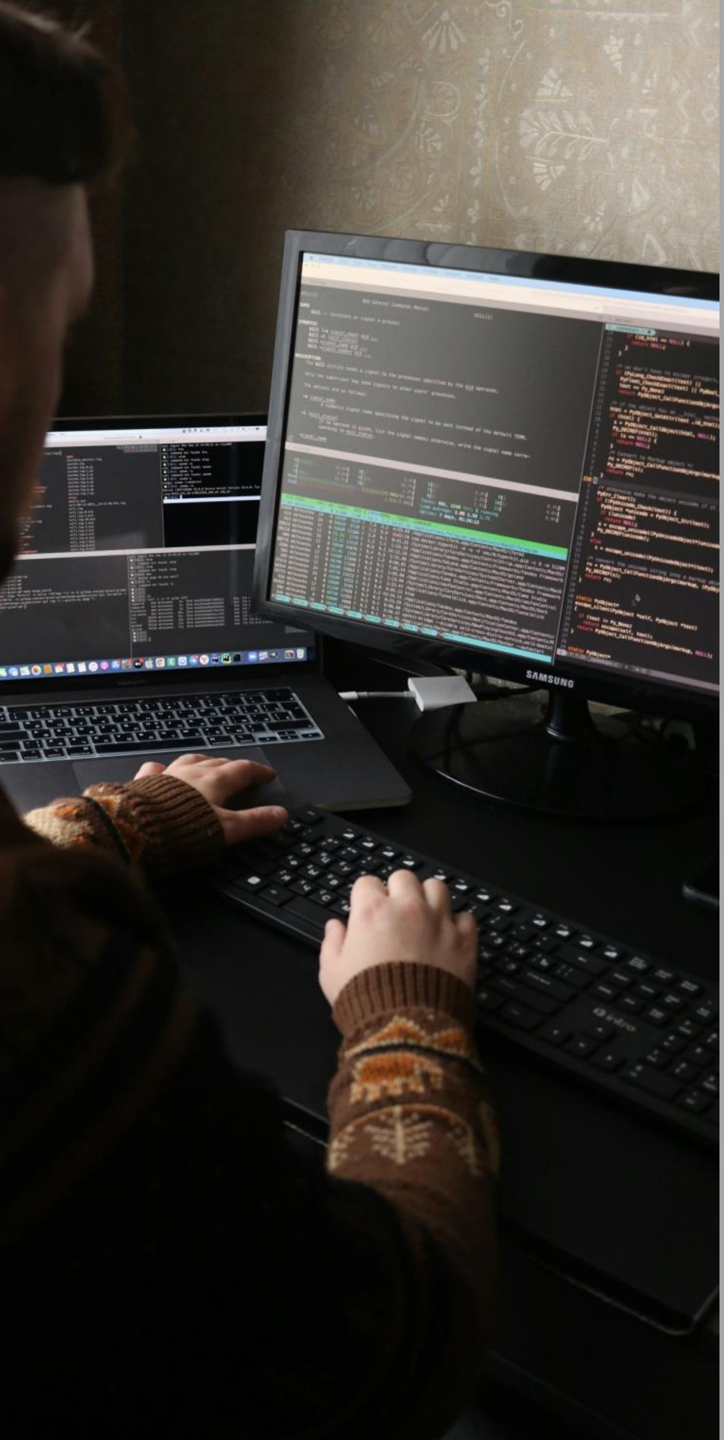


Week 7
Lecture Note

Parsing and Creating XML Documents with DOM, SAX

Feb 26-March 1, 20

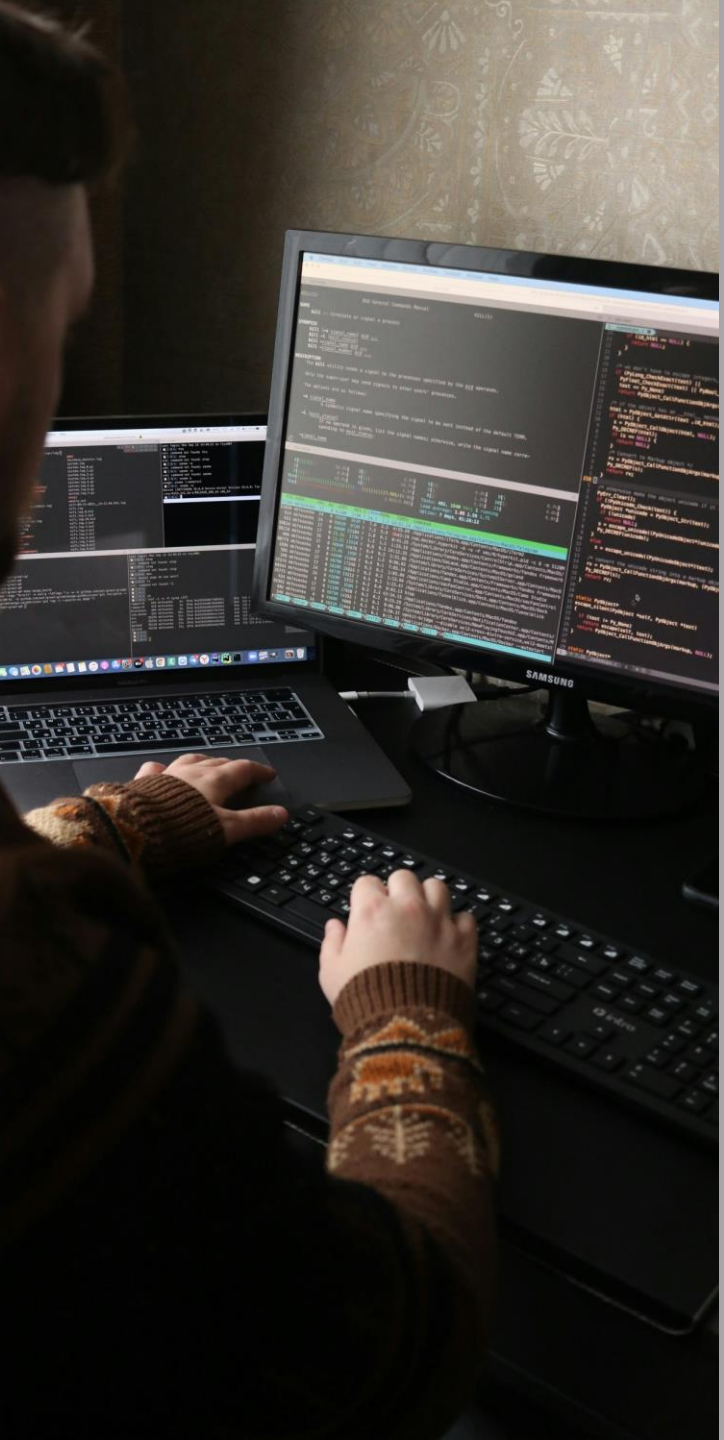


Week 7
Lecture Note

Learning Objectives:

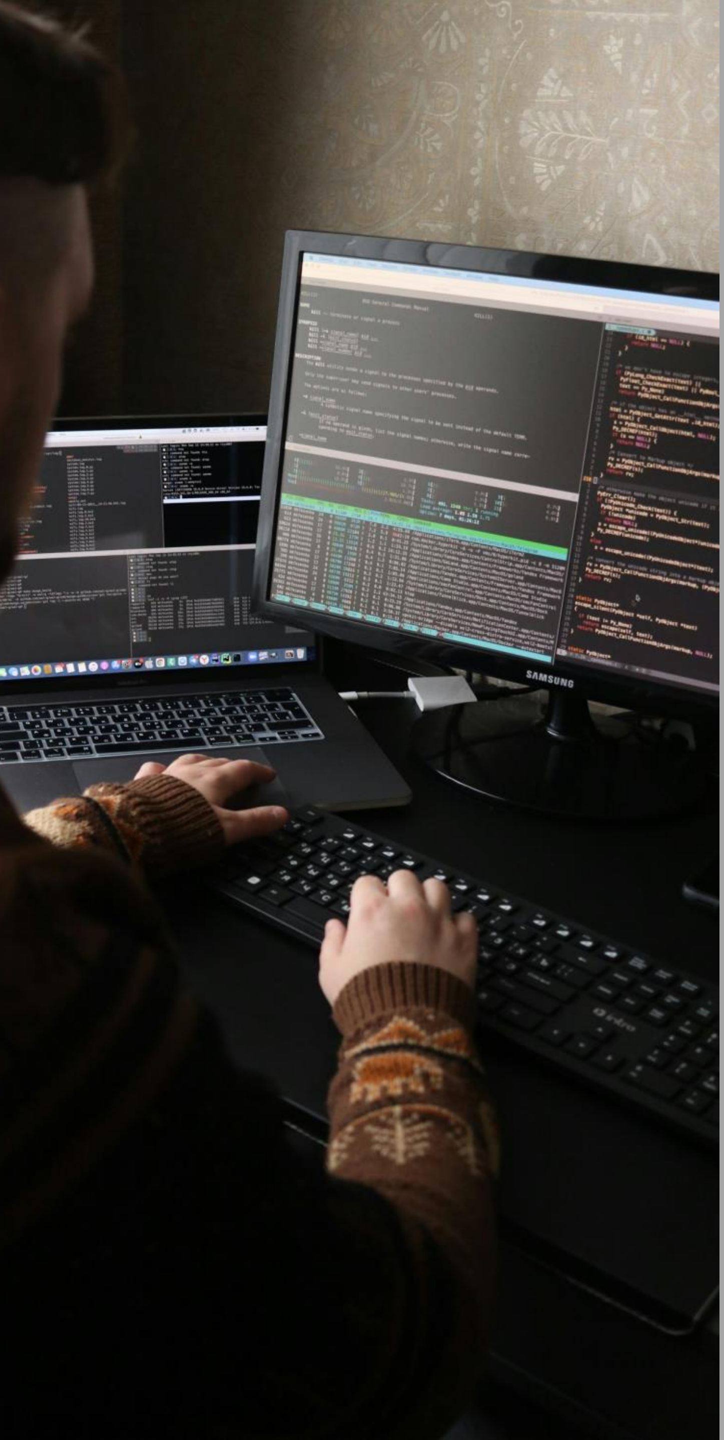
Introduce the XML DOM (Document Object Model).
Construct an XML DOM parser script.

Feb 26-March 1, 20



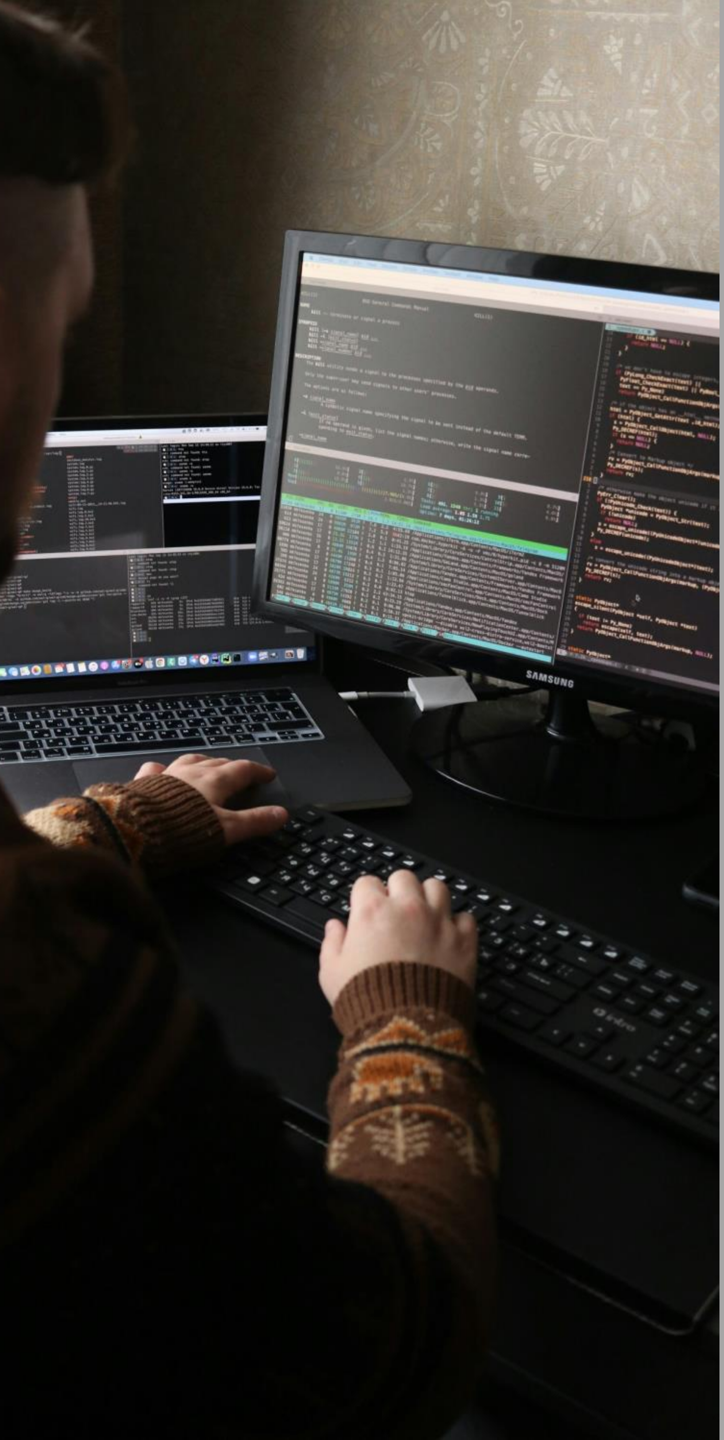
What Is DOM?

Document Object Model (DOM) is a Java
API for parsing an XML document into an
in-memory tree of nodes and for creating
an XML document from a node tree.
After a DOM parser creates a tree, an
application uses the DOM API to navigate
over and extract infoset(dataset) items
from the tree's nodes.



A Tree of Nodes.

DOM views an XML document as a tree that's composed of several kinds of nodes. This tree has a single *root node*, and all nodes except for the root have a *parent node*. Also, each node has a *list of child nodes*. When this list is empty, the child node is known as a *leaf node*.

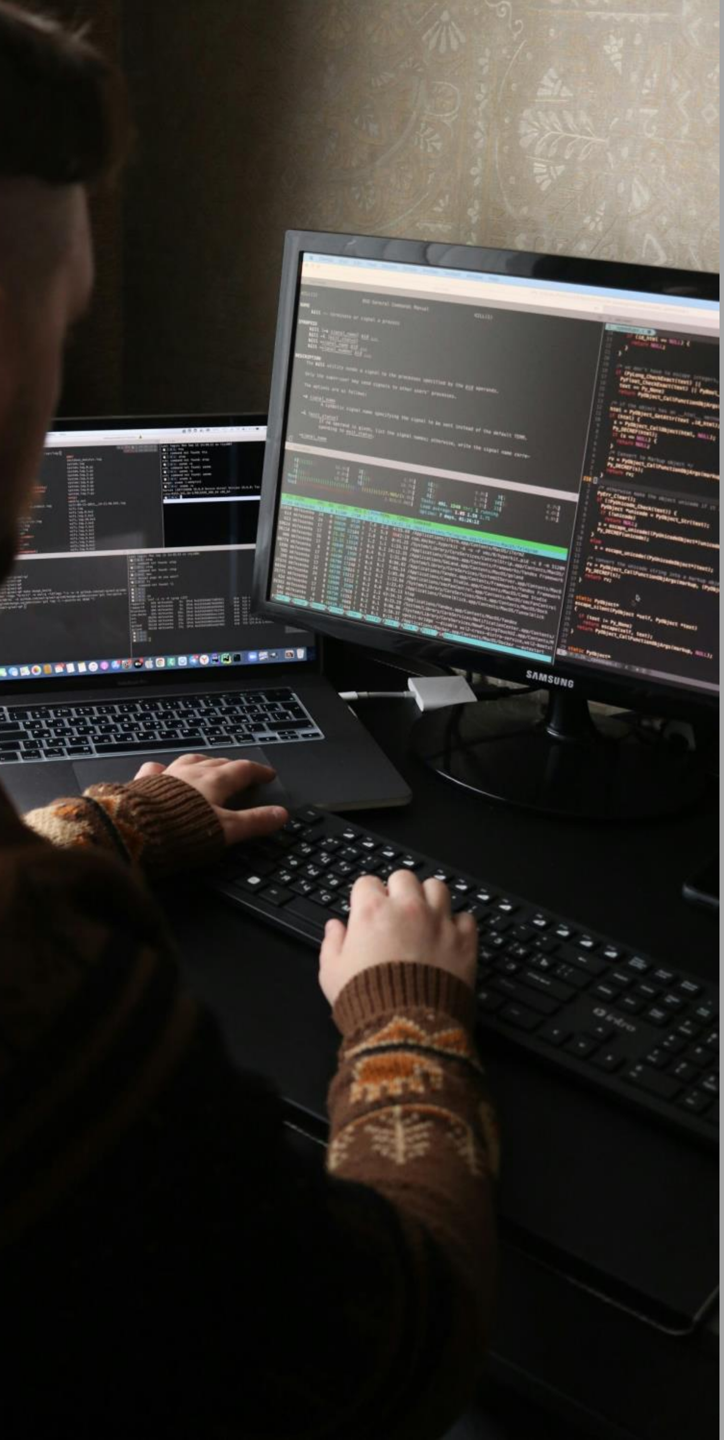


Exploring the DOM API

Java implements DOM through the *javax.xml.parsers* package's abstract

DocumentBuilder and DocumentBuilderFactory classes

The *org.w3c.dom* packages provide various types that augment this implementation.

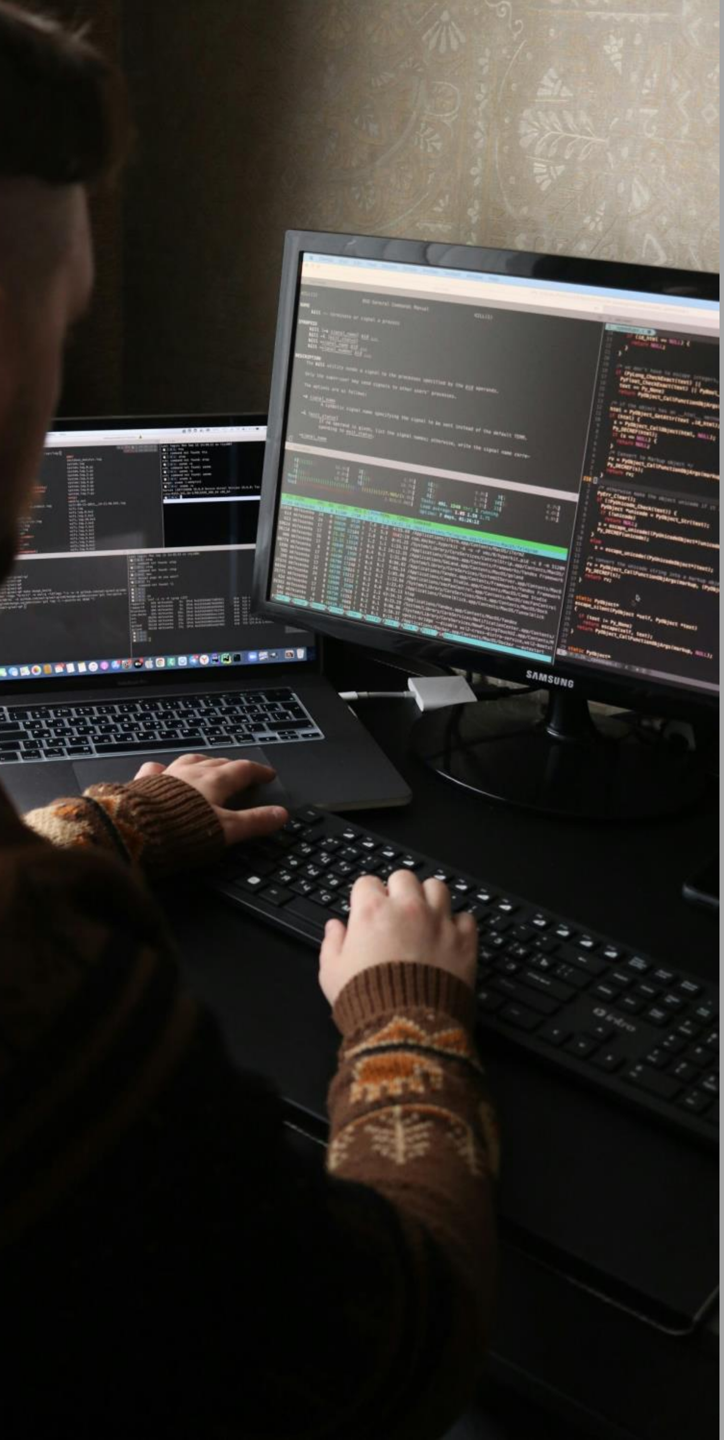


Obtaining a DOM Parser /Document Builder

A *DOM parser* is also known as a *document builder* because of its dual role in parsing and creating XML documents.

You obtain a DOM parser/document builder by first instantiating

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



Obtaining a DOM Parser /Document Builder

You obtain a DOM parser/document builder by first instantiating

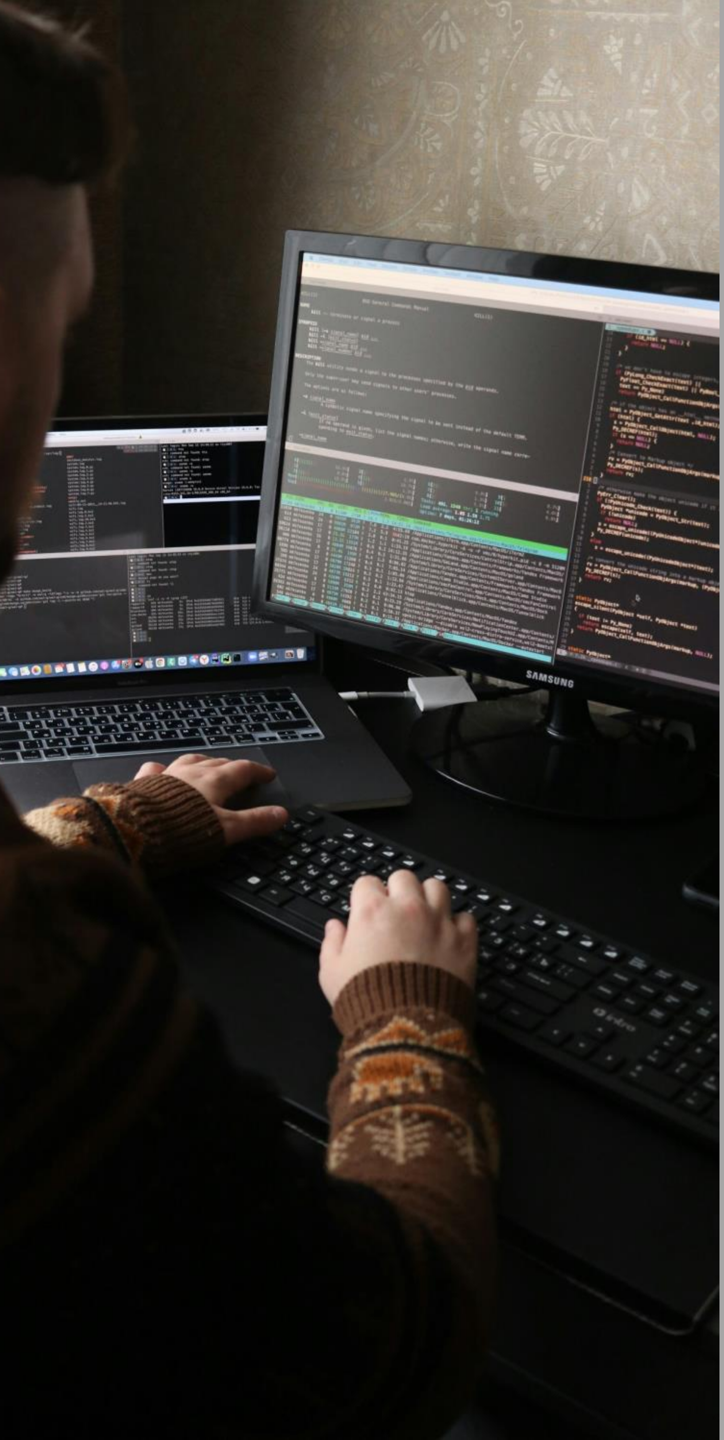
```
DocumentBuilderFactory dbf = DocumentBuilderFactory
                                     .newInstance();
```

//After the factory has been configured, call its DocumentBuilder to return a document builder that supports the configuration

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

OR simple use this

```
DocumentBuilder docBuild = DocumentBuilderFactory
                                    .newInstance()
                                    .newDocumentBuilder();
```



Parsing XML Documents

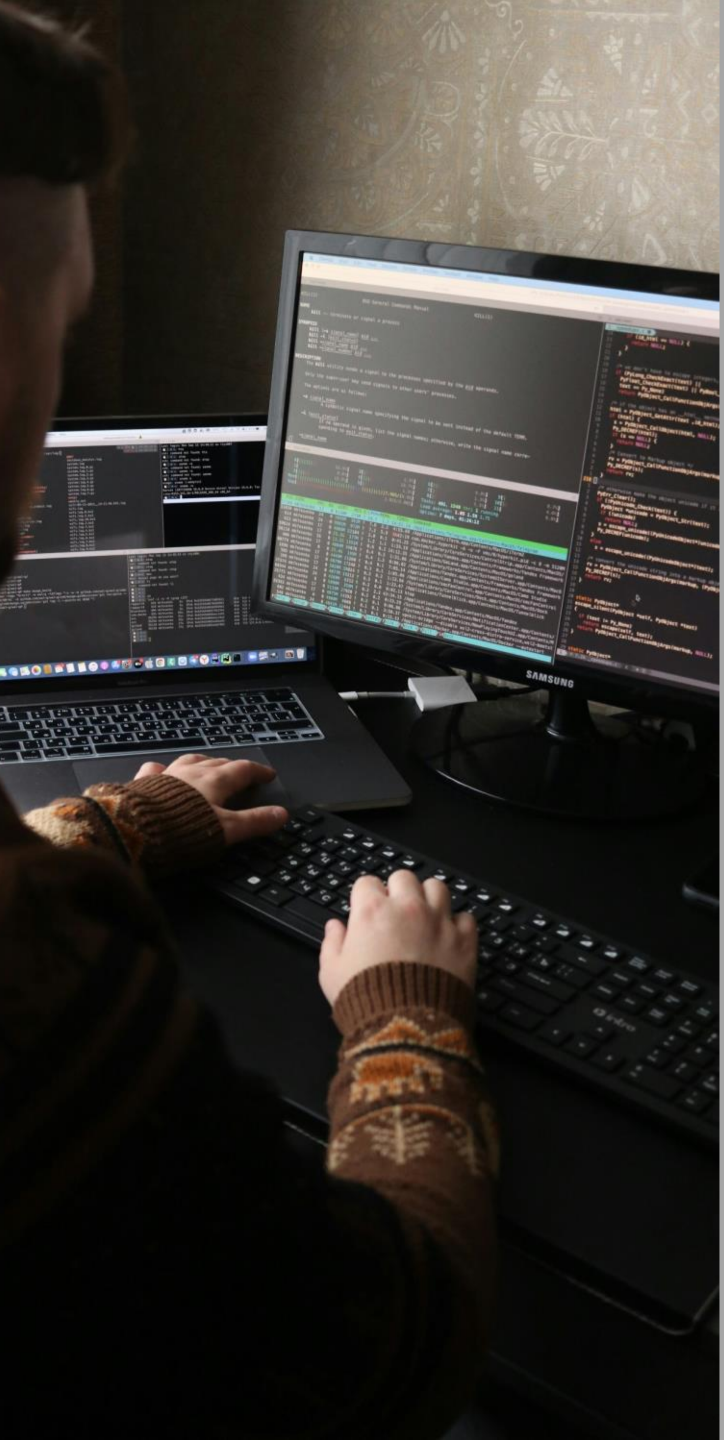
Assuming that you've successfully obtained a document builder what happens next depends on whether you want to **parse** or **create** an XML document.

DocumentBuilder also declares the abstract Document newDocument() method for creating a DOM tree.

Document and all other *org.w3c.dom* interfaces that describe different kinds of nodes are subinterfaces of the org.w3c.dom.Node interface. As such, they inherit Node's constants and methods.

Document declares methods for locating one or more elements:

- Element getElementById(String elementId) returns the element that has an id attribute (<elem id=…>) matching the value specified tagName.
- NodeList getElementsByTagName(String tagname) returns a nodelist of a document's elements (in document order) matching the specified tagName.



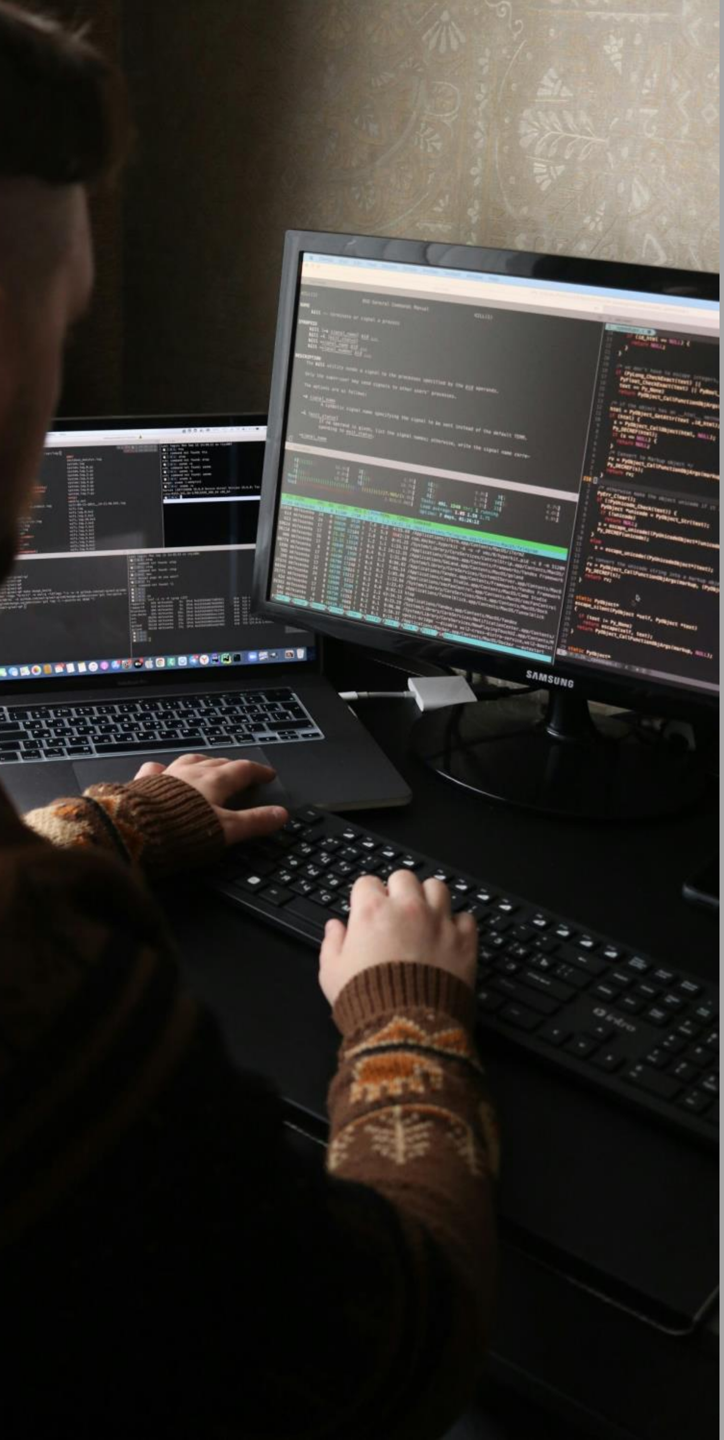
```java
import java.io.File;

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

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;


public class DOMSample {
  public static void main(String args[]){
    try {

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

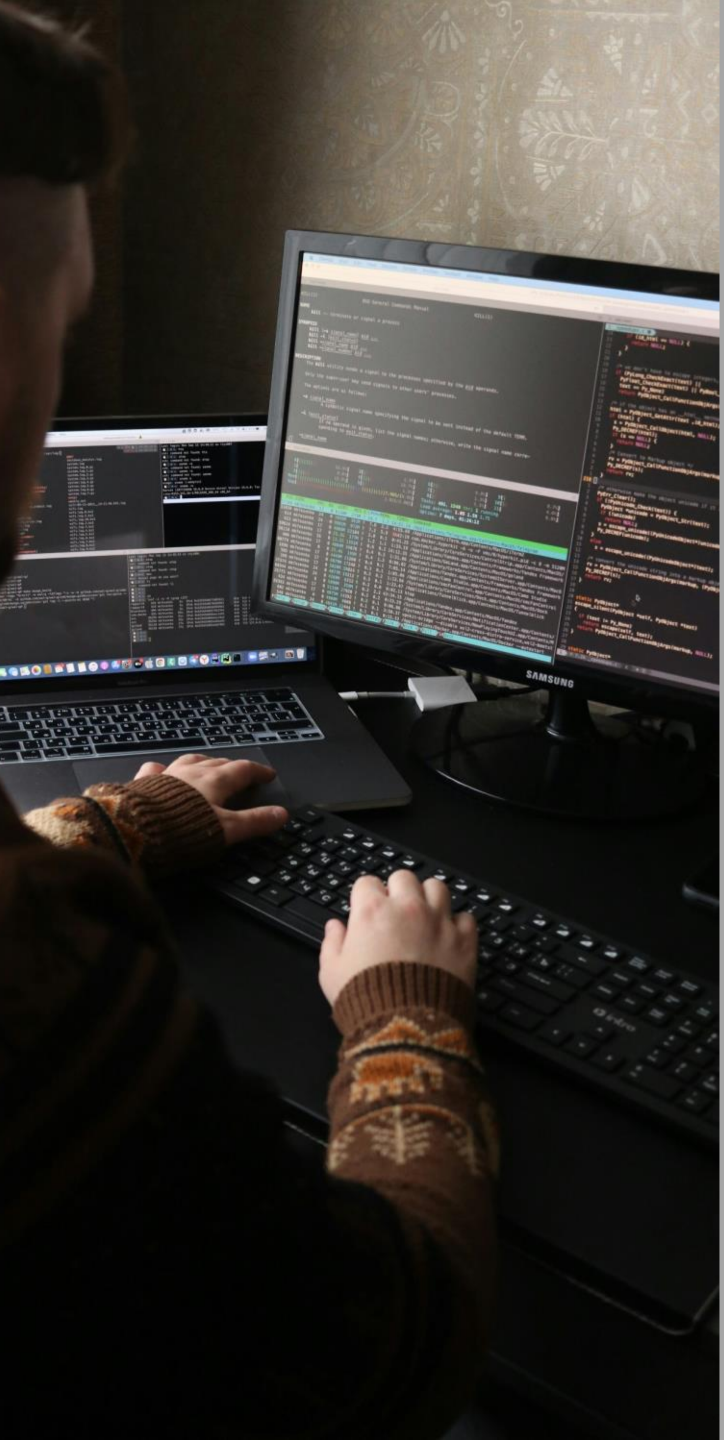


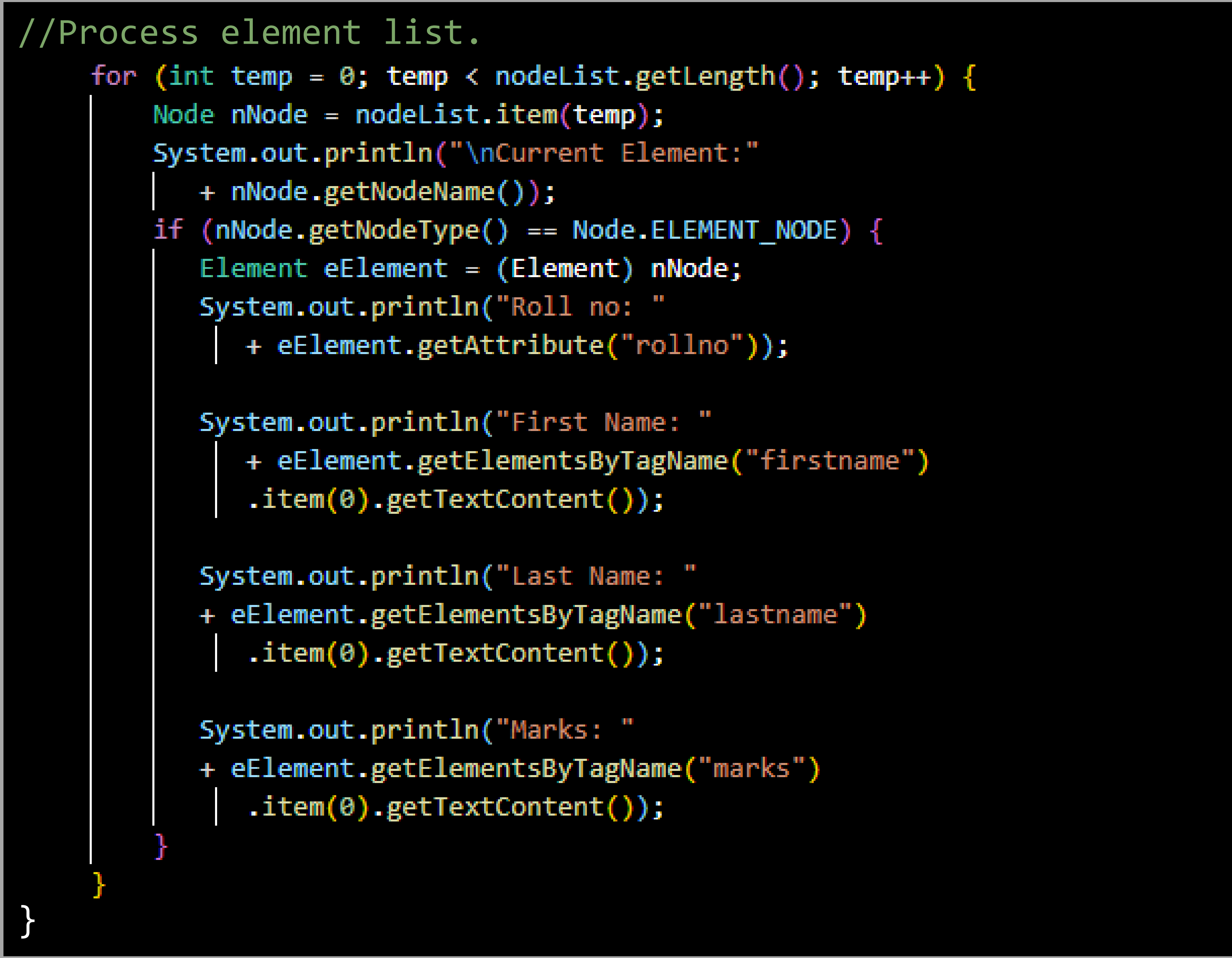
```
try {
        DocumentBuilderFactory dbFactory =
                    DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder =
                    dbFactory.newDocumentBuilder();

        Document document =
                dBuilder.parse(new File("StudentTest.txt"));
        document.getDocumentElement().normalize();


//Print root element.
        System.out.println("Root element:"
        + document.getDocumentElement().getNodeName());

//Get element list.
        NodeList nodeList =
              document.getElementsByTagName("student");
```



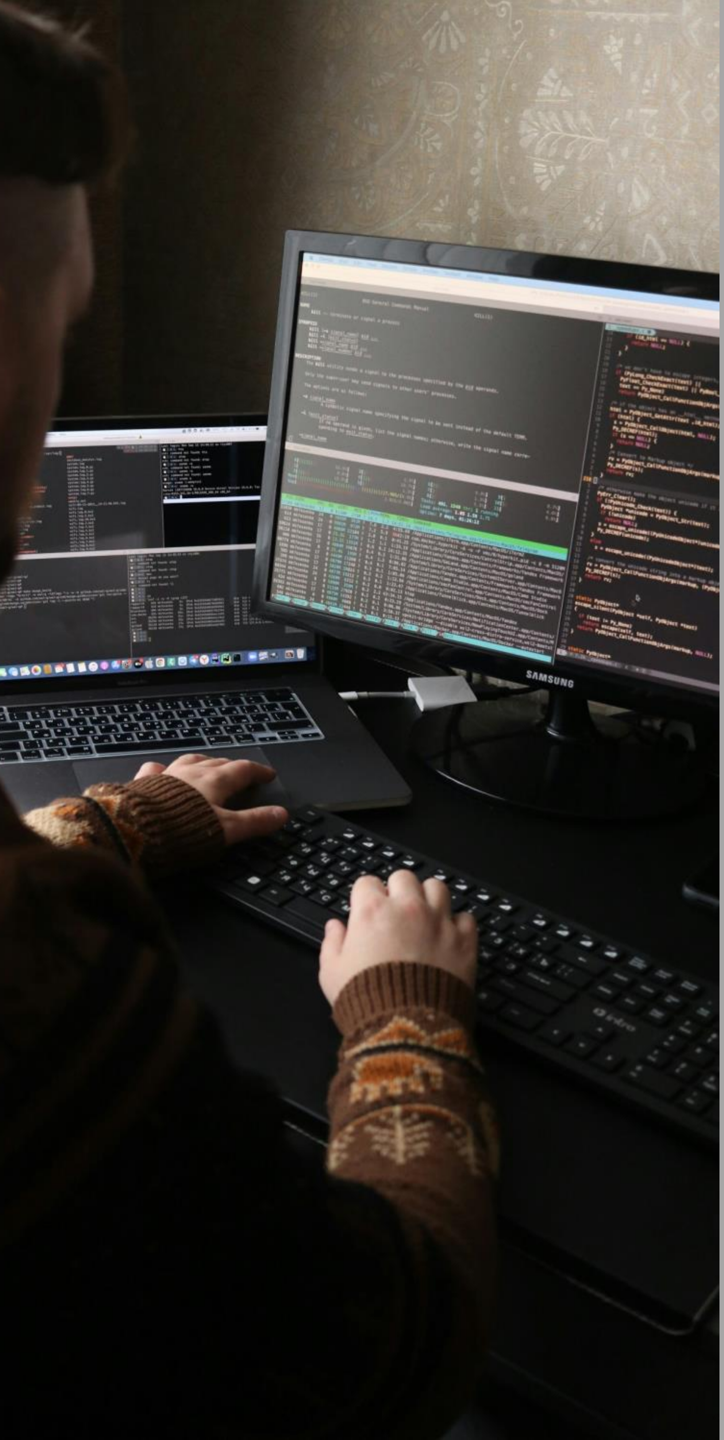


```
//Process element list.
    for (int temp = 0; temp < nodeList.getLength(); temp++) {
        Node nNode = nodeList.item(temp);
        System.out.println("\nCurrent Element:"
            + nNode.getNodeName());
        if (nNode.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) nNode;
            System.out.println("Roll no: "
                + eElement.getAttribute("rollno"));

            System.out.println("First Name: "
                + eElement.getElementsByTagName("firstname")
                .item(0).getTextContent());

            System.out.println("Last Name: "
            + eElement.getElementsByTagName("lastname")
                .item(0).getTextContent());

            System.out.println("Marks: "
            + eElement.getElementsByTagName("marks")
                .item(0).getTextContent());
        }
    }
}
```



```
//Process element list.
    for (int temp = 0; temp < nodeList.getLength(); temp++) {
        Node nNode = nodeList.item(temp);
        System.out.println("\nCurrent Element:"
            + nNode.getNodeName());
        if (nNode.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) nNode;
            System.out.println("Roll no: "
                + eElement.getAttribute("rollno"));

            System.out.println("First Name: "
                + eElement.getElementsByTagName("firstname")
                .item(0).getTextContent());

            System.out.println("Last Name: "
            + eElement.getElementsByTagName("lastname")
                .item(0).getTextContent());

            System.out.println("Marks: "
            + eElement.getElementsByTagName("marks")
                .item(0).getTextContent());
        }
    }
}
```