

Нехай існує деякий XML-документ (input.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<order id="1">
  <!--Важливе замовлення для Івасика-Телесика -->
  <person>Івасик-Телесик</person>
  <item>
    <title>Гуси-лебеді</title>
    <quantity>5</quantity>
  </item>
  <item>
    <title>Яблука</title>
    <quantity>10</quantity>
  </item>
</order>
```

Скористаємося DOM-парсером для отримання доступу до елементів та атрибутів:

```
import java.io.File;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.*;
public class dom {
    public static void main(String[] args) {
        try {
            File inputFile = new File("input.xml");
            DocumentBuilderFactory dbFactory
                = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();
            System.out.println("Кореневий елемент: "
                + doc.getDocumentElement().getNodeName());
            NodeList nList = doc.getDocumentElement().getChildNodes();
            System.out.println("Кількість дочірніх вузлів: " + nList.getLength());
            System.out.println("-----");
            for (int i = 0, l = nList.getLength(); i < l; i++) {
                Node nNode = nList.item(i);
                System.out.print("Child node #" + i + " - " + nNode.getNodeName() + ": ");
                System.out.println(nNode.getTextContent());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Результат:

Кореневий елемент: order  
Кількість дочірніх вузлів: 9

-----  
Child node #0 - #text:

Child node #1 - #comment: Важливе замовлення для Івасика-Телесика  
Child node #2 - #text:

Child node #3 - person: Івасик-Телесик  
Child node #4 - #text:

Child node #5 - item:  
    Гуси-лебеді  
    5

Child node #6 - #text:

Child node #7 - item:  
    Яблука  
    10

Зверніть увагу на те, що як відступи у xml-документові використано табуляції. Замість табуляції можна використати для відступів 4 (загальноприйняте значення, за потреби можна встановити іншу кількість) послідовних пробіли. Для зміни способу організації відступів скористайтесь меню Edit → Convert Indents → To Spaces/To Tabs. Налаштувати відступи можна з меню File → Settings → Editor → Code Style → XML (<https://www.jetbrains.com/idea/help/code-style-xml.html#d1227290e130>).

Приберемо із input.xml форматування, записавши його одним рядком:

```
<?xml version="1.0" encoding="UTF-8"?><order id="1"><!--Важливе замовлення для Івасика-Телесика --><person>Івасик-Телесик</person><item><title>Гуси-лебеді</title><quantity>5</quantity></item><item><title>Яблука</title><quantity>10</quantity></item></order>
```

Результат відрізнятиметься від отриманого щойно:

Кореневий елемент: order  
Кількість дочірніх вузлів: 4

-----  
Child node #0 - #comment: Важливе замовлення для Івасика-Телесика

Child node #1 - person: Івасик-Телесик

Child node #2 - item: Гуси-лебеді5

Child node #3 - item: Яблука10

Легко помітити, що результати відрізняються наявністю вузлів #text. У даному випадкові ці вузли містять символ переходу на новий рядок. Змініть документ таким чином, щоб він містив наступне: `<order id="1">hello<!--` . Результат включатиме вузол #text:

Кореневий елемент: order

Кількість дочірніх вузлів: 5

-----

Child node #0 - #text: hello

Child node #1 - #comment: Важливе замовлення для Івасика-Телесика

Child node #2 - person: Івасик-Телесик

Child node #3 - item: Гуси-лебеді5

Child node #4 - item: Яблука10

#text — це результат виконання `getNodeName()` над порожнім вузлом. Позбутися таких вузлів можна різними способами. Модифікуємо код прикладу:

```
for (int i = 0, l = nList.getLength(); i < l; i++) {
    Node nNode = nList.item(i);
    if(nNode instanceof Text) {
        String value = nNode.getNodeValue().trim();
        if (value.equals("")) continue;
    }
    System.out.print("Child node #" + i + " - " + nNode.getNodeName() + ": ");
    System.out.println(nNode.getTextContent());
}
```

Отримаємо:

Кореневий елемент: order

Кількість дочірніх вузлів: 9

-----

Child node #1 - #comment: Важливе замовлення для Івасика-Телесика

Child node #3 - person: Івасик-Телесик

Child node #5 - item:

Гуси-лебеді

5

Child node #7 - item:

Яблука

10

Якщо потрібно розглянути тільки дочірні елементи, можна скористатися наступним підходом:

```
for (int i = 0, l = nList.getLength(); i < l; i++) {
    Node nNode = nList.item(i);
    if(nNode.getNodeType() == nNode.ELEMENT_NODE) {
        System.out.print("Child node #" + i + " - " + nNode.getNodeName() + ": ");
        System.out.println(nNode.getTextContent());
    }
}
```

У цьому випадкові взято до уваги тільки дочірні вузли — елементи. Вузли іншого типу проігноровані.  
Значення nodeName, nodeValue та attributes залежать від типу вузла. Детальніше: <https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/Node.html>.

Interface	nodeName	nodeValue	атрибути
Attr	Те саме, що і Attr.name	Те саме, що і Attr.value	null
CDataSection	"#cdata-section"	Те саме, що і CharacterData.data, вміст секції CDATA	null
Comment	"#comment"	Те саме, що і CharacterData.data, вміст коментаря	null
Document	"#document"	null	null
DocumentFragment	"#document-fragment"	null	null
DocumentType	Те саме, що і DocumentType.name	null	null
Element	Те саме, що і Element.tagName	null	Named NodeMap
Entity	Ім'я сутності	null	null
EntityReference	Ім'я сутності, на яку посилаються	null	null
Notation	Ім'я нотації	null	null

Interface	nodeName	nodeValue	атрибути
Processing Instruction	Те саме, що і ProcessingInstruction.target	Те саме, що і ProcessingInstruction.data	null
Text	"#text"	Те саме, що і CharacterData.data, вміст текстового вузла	null

Метод [getTextContent\(\)](#) повертає текстовий вміст даного вузла та його нащадків. Повернутий рядок не містить розмітки. Результат залежить від типу вузла.

Тип вузла	Коментар
ELEMENT_NODE, ATTRIBUTE_NODE, ENTITY_NODE, ENTITY_REFERENCE_NODE, DOCUMENT_FRAGMENT_NODE	поєднання значення атрибута textContent кожного дочірнього вузла за винятком вузлів COMMENT_NODE та PROCESSING_INSTRUCTION_NODE. Якщо вузол не має нащадків, повертається пустий рядок.
TEXT_NODE, CDATA_SECTION_NODE, COMMENT_NODE, PROCESSING_INSTRUCTION_NODE	nodeValue
DOCUMENT_NODE, DOCUMENT_TYPE_NODE, NOTATION_NODE	null

У наступному прикладі отримують доступ до значень атрибутів елемента. Скористаємося методом [getAttributes\(\)](#), для цього дещо модифікуємо XML-документ:

```
<?xml version="1.0" encoding="UTF-8"?>
<order id="1">
  <!--Важливе замовлення для Івасика-Телесика -->
  <person>Івасик-Телесик</person>
  <item id="1" status="delivered">
    <title>Гуси-лебеді</title>
    <quantity>5</quantity>
  </item>
  <item id="2" status="delivered">
    <title>Яблука</title>
    <quantity>10</quantity>
  </item>
</order>
```

Вище до елементів <item> додано атрибути id та status. Зверніть увагу, що відступи організовано за допомогою пробілів (1 відступ — 4 пробіли). На роботу парсера це не впливає.

```

import java.io.File;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.*;
public class dom {
    public static void main(String[] args) {
        try {
            File inputFile = new File("input.xml");
            DocumentBuilderFactory dbFactory
                = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();
            System.out.println("Кореневий елемент: "
                + doc.getDocumentElement().getNodeName());
            NodeList nList = doc.getDocumentElement().getChildNodes();
            System.out.println("Кількість дочірніх вузлів: " + nList.getLength());
            System.out.println("-----");
            for (int i = 0, l = nList.getLength(); i < l; i++) {
                Node nNode = nList.item(i);
                if(nNode.getNodeType() == nNode.ELEMENT_NODE) {
                    System.out.println("Child node #" + i + " - " + nNode.getNodeName());
                    NamedNodeMap chNodes = nNode.getAttributes();
                    for (int j = 0, len = chNodes.getLength(); j < len; j++) {
                        Node chNode = chNodes.item(j);
                        System.out.println(chNode.getNodeName() + ": " +
                            chNode.getTextContent());
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Результат:

```

Кореневий елемент: order
Кількість дочірніх вузлів: 9
-----
Child node #3 - person
Child node #5 - item
id: 1
status: delivered
Child node #7 - item
id: 2
status: delivered

```

Нехай дано XML-документ:

```
<?xml version="1.0" encoding="UTF-8"?>
<order id="1">
  <!--Важливе замовлення для Івасика-Телесика -->
  <person>Івасик-Телесик</person>
  <item id="1" status="delivered">
    <title>Гуси-лебеді</title>
    <quantity>5</quantity>
  </item>
  <item id="2" status="delivered">
    <title>Яблука</title>
    <quantity>10</quantity>
  </item>
</order>
```

Завдання полягає у тому, щоб знайти у ньому елементи `<item>`, їхні атрибути та відповідні дочірні елементи і їхні текстові значення.

```
import java.io.File;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.*;
public class dom {
  public static void main(String[] args) {
    try {
      File inputFile = new File("input.xml");
      DocumentBuilderFactory dbFactory
        = DocumentBuilderFactory.newInstance();
      DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
      Document doc = dBuilder.parse(inputFile);
      doc.getDocumentElement().normalize();
      System.out.println("Кореневий елемент: "
        + doc.getDocumentElement().getNodeName());
      NodeList nList = doc.getElementsByTagName("item");
      // якщо потрібно вибрати усі елементи, скористайтесь
      // параметром методу getElementsByTagName("*");
      System.out.println("Кількість елементів <item>: " + nList.getLength());
      System.out.println("-----");
      for (int i = 0, l = nList.getLength(); i < l; i++) {
        Node nNode = nList.item(i);
        Element element = (Element) nNode;
        String id = element.getAttribute("id");
        String status = element.getAttribute("status");
        System.out.println("id: " + id);
        System.out.println("status: " + status);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

```

        NodeList childTitleElements = element.getElementsByTagName("title");
        String childTitle = childTitleElements.item(0).getTextContent();
        System.out.println("title: " + childTitle);
        NodeList childQuantityElements = element.getElementsByTagName("quantity");
        String childQuantity = childQuantityElements.item(0).getTextContent();
        System.out.println("quantity: " + childQuantity);
        System.out.println("-----");
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

Результат:

```

Кореневий елемент: order
Кількість елементів <item>: 2
-----
id: 1
status: delivered
title: Гуси-лебеді
quantity: 5
-----
id: 2
status: delivered
title: Яблука
quantity: 10
-----

```