

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ В. Н. КАРАЗІНА  
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ

ЛАБОРАТОРНА РОБОТА № 3  
З дисципліни «Крос-платформне програмування»  
з теми “ Java & XML ”

Виконав:

студент 2 курсу, групи КС24

Бондаренко Олег

Перевірив:

Споров Олександр Євгенович

## Завдання №0

Із сайту з відкритими даними (<https://catalog.data.gov/dataset/popular-baby-names>) було отримано свіжий (від 3 березня, 2023), великий за розміром датасет в XML форматі з інформацією про популярні імена дітей у місті Нью-Йорк. Цей датасет складений за офіційною інформацією із служби реєстрації актів цивільного стану міста Нью-Йорка. Архів з цим датасетом має назву **Popular\_Baby\_Names\_NY.zip** та розміщений в лекційному Гугл-класі в розділі **Методичні вказівки з виконання лабораторних робіт**. Кожен запис цього датасету представляє інформацію про дитину: вказано дату народження, гендер, етнічну приналежність мами, власне ім'я дитини, кількість (*count*) дітей з цим іменем та рейтинг (*rating*) імені у відповідній групі. Потрібно провести попередній аналіз цих даних та вибрати з них лише потрібну для подальшої роботи інформацію. Виконати наступні завдання:

- ◆ Написати програму для виведення на екран частини XML документу за допомогою SAX парсеру без валідації для вивчення його структури та вмісту; програмно отримати перелік всіх тегів, імена яких присутні в документі.
- ◆ За невеликим характерним фрагментом скласти xsd схему документу, створити валідатор та перевірити, чи правильно було зрозуміло структуру документу.
- ◆ Написати програмне рішення, що за допомогою SAX парсеру без валідації отримає назви всіх національних груп, що представлені в документі.
- ◆ Написати додаток, що з всього XML документу вибирає задану кількість найбільш популярних імен в заданій етнічній групі із зберіганням інформації про: ім'я, гендер, кількість імен та рейтинг імен, а також створює відповідні Java об'єкти для зберігання цієї інформації та сортує інформацію по збільшенню номеру в рейтингу. Зберегти

вибрану та відсортовану інформацію до нового XML файлу за допомогою DOM парсеру.

- ◆ Прочитати цей новий документ за допомогою DOM парсеру та вивести інформацію, що в ньому зберігається, на екран.

### 1. Напишемо аналізатор XML із допомогою SAX парсеру:

Лістинг 1 код SAX парсеру:

```
import org.xml.sax.SAXException;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import java.io.File;
import java.io.IOException;

public class Main {
    public static void main(String[] args) throws
        ParserConfigurationException, SAXException, IOException {
        SAXParser saxParser =
            SAXParserFactory.newInstance().newSAXParser();
        DataHandler handler = new DataHandler();
        saxParser.parse(new File("Popular_Baby_Names_NY.xml"), handler);
        System.out.println("\n\nAll TAGS");
        for (String str: handler.getTags()) {
            if(str!=null) System.out.println(str);
        }
    }
}
```

```

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

class DataHandler extends DefaultHandler {
    public String[] getTags() {
        return tags;
    }

    private String[] tags;
    private int cnt = 0;

    public DataHandler() {
        tags = new String[10];
    }

    @Override
    public void startDocument() throws SAXException {
        super.startDocument();
    }

    @Override
    public void endDocument() throws SAXException {
        super.endDocument();
    }

    @Override
    public void startElement(String uri, String localName, String qName,
Attributes attributes) throws SAXException {
        System.out.println(qName);
        for (int i = 0; i < cnt; i++) {
            if (tags[i].compareTo(qName) == 0) {
                return;
            }
        }
        tags[cnt++] = qName;
    }

    @Override
    public void endElement(String uri, String localName, String qName)
throws SAXException {
        System.out.println(qName);
    }

    @Override
    public void characters(char[] ch, int start, int length) throws
SAXException {
        String info = new String(ch, start, length);
        info = info.replace("\n", "").trim();
        if (!info.isEmpty()) {
            System.out.println("\t" + info.trim());
        }
    }
}

```

```

row
row
brth_yr
    2018
brth_yr
gndr
    FEMALE
gndr
ethcty
    BLACK NON HISPANIC
ethcty
nm
    Zahra
nm
cnt
    10
cnt
rnk
    40
rnk
row
row
response

```

Рисунок 1. Частина зчитаного файлу

```

ALL TAGS
response
row
brth_yr
gndr
ethcty
nm
cnt
rnk

```

Рисунок 2. Теги

2. Було створено за документом .xsd схему.

```

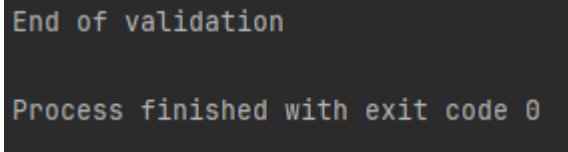
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" targetNamespace="">
  <xsd:element name="response">
    <xsd:complexType mixed="true">
      <xsd:sequence>
        <xsd:element minOccurs="0" name="row">
          <xsd:complexType mixed="true">
            <xsd:sequence>
              <xsd:element maxOccurs="unbounded" name="row">

```

```

                                <xsd:complexType mixed="true">
                                    <xsd:sequence>
                                        <xsd:element minOccurs="0"
name="brth_yr" type="xsd:int"/>
                                        <xsd:element minOccurs="0"
name="gnldr" type="xsd:normalizedString"/>
                                        <xsd:element minOccurs="0"
name="ethcty" type="xsd:normalizedString"/>
                                        <xsd:element minOccurs="0"
name="nm" type="xsd:normalizedString"/>
                                        <xsd:element minOccurs="0"
name="cnt" type="xsd:int"/>
                                        <xsd:element minOccurs="0"
name="rnk" type="xsd:int"/>
                                    </xsd:sequence>
                                    <xsd:attribute name="_id"
type="xsd:normalizedString" use="required"/>
                                    <xsd:attribute name="_uuid"
type="xsd:normalizedString" use="required"/>
                                    <xsd:attribute name="_position"
type="xsd:int" use="required"/>
                                    <xsd:attribute name="_address"
type="xsd:anyURI" use="required"/>
                                </xsd:complexType>
                            </xsd:element>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>

```



```

End of validation

Process finished with exit code 0

```

Рисунок 3. Успішна валідація

## Лістинг 2 код валідатора:

```

package valid;

import org.xml.sax.SAXException;

import javax.xml.XMLConstants;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import java.io.File;
import java.io.IOException;

public class Validator {

    public static final String filePathXSD = "xml-schema.xsd";

    public static final String filePathXML = "Popular_Baby_Names_NY.xml";
    public static void main(String[] args) throws SAXException,
ParserConfigurationException, IOException {

```

```

        SchemaFactory sf =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
        Schema schema = sf.newSchema(new File(filePathXSD));

        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

        dbf.setSchema(schema);
        dbf.setValidating(false);
        dbf.setNamespaceAware(true);
        dbf.setIgnoringElementContentWhitespace(true);

        File xmlFile = new File(filePathXML);

        DocumentBuilder builder = dbf.newDocumentBuilder();
        builder.setErrorHandler(new SimpleErrorHandler());
        builder.parse(xmlFile);

        System.out.println("End of validation");
    }
}

```

3. Напишемо програму, теги додаються до ліста, потім переформатовуються у Set, щоб знищити дублікати.

Лістинг 3:

```

package finder;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import java.io.File;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Set;

public class Main {
    public static ArrayList<String> ethnicity = new ArrayList<>();
    public static void main(String[] args) throws
ParserConfigurationException, SAXException, IOException {
        DefaultHandler defaultHandler = new DefaultHandler() {
            String currentTag;
            @Override
            public void startDocument() throws SAXException {
                super.startDocument();
            }

            @Override
            public void endDocument() throws SAXException {
                super.endDocument();
            }

            @Override
            public void startElement(String uri, String localName, String
qName, Attributes attributes) throws SAXException {
                currentTag = qName;
            }

            @Override

```

```

        public void endElement(String uri, String localName, String
qName) throws SAXException {
            super.endElement(uri, localName, qName);
        }
        @Override
        public void characters(char[] ch, int start, int length) throws
SAXException {
            if(currentTag.contains("ethcty")){
                Main.ethnicity.add(new String(ch,start,length));
            }
        }
    };

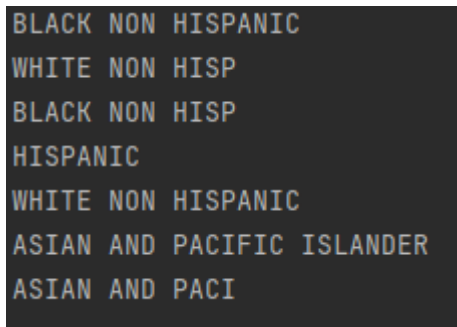
    SAXParser saxParser =
SAXParserFactory.newInstance().newSAXParser();

    saxParser.parse(new
File("Popular_Baby_Names_NY.xml"),defaultHandler);

    Set<String> set = new HashSet<>(ethnicity);

    set.stream().forEach(System.out::println);
}
}

```



```

BLACK NON HISPANIC
WHITE NON HISP
BLACK NON HISP
HISPANIC
WHITE NON HISPANIC
ASIAN AND PACIFIC ISLANDER
ASIAN AND PACI

```

Рисунок 4. Види етнічності

### 3. Зчитуємо xml-документ дом парсером:

Лістинг 4:

```

package DOM;

import org.w3c.dom.CharacterData;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.*;
import java.nio.charset.StandardCharsets;

public class DOM_parser {
    private Document m_doc;

```

```

        public DOM_parser(String xmlfile) throws Exception {
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            m_doc = builder.parse(new FileInputStream(new File(xmlfile)));
        }
        public DOM_parser() throws Exception {
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            m_doc = builder.newDocument();
        }

        public int getChildCount(String parentTag, int parentIndex, String
childTag) {
            NodeList list = m_doc.getElementsByTagName(parentTag);
            Element parent = (Element) list.item(parentIndex);
            NodeList childList = parent.getElementsByTagName(childTag);
            return childList.getLength();
        }

        public String getChildValue(String parentTag, int parentIndex, String
childTag, int childIndex) {
            NodeList list = m_doc.getElementsByTagName(parentTag);
            Element parent = (Element) list.item(parentIndex);
            NodeList childList = parent.getElementsByTagName(childTag);
            Element element = (Element) childList.item(childIndex);
            Node child = element.getFirstChild();
            if (child instanceof CharacterData) {
                CharacterData cd = (CharacterData) child;
                return cd.getData();
            }
            return "";
        }

        public String getChildAttribute(String parentTag, int parentIndex, String
childTag, int childIndex,
                                     String attributeTag) {
            NodeList list = m_doc.getElementsByTagName(parentTag);
            Element parent = (Element) list.item(parentIndex);
            NodeList childList = parent.getElementsByTagName(childTag);
            Element element = (Element) childList.item(childIndex);
            return element.getAttribute(attributeTag);
        }

        public int getRootElementCount(String rootElementTag) {
            NodeList list = m_doc.getElementsByTagName(rootElementTag);
            return list.getLength();
        }

        public void addChildElement(String parentTag, int parentIndex, String
childTag, String childValue) {
            NodeList list = m_doc.getElementsByTagName(parentTag);
            Element parent = (Element) list.item(parentIndex);
            Element child = m_doc.createElement(childTag);
            if (childValue != null) {
                child.appendChild(m_doc.createTextNode(childValue));
            }
            if (parent == null) {
                m_doc.appendChild(child);
            }
            else {
                parent.appendChild(child);
            }
        }

```



```

    public void setAttributeValue(String elementTag, int elementIndex, String
attributeTag,
                                String attributeValue) {
        NodeList list = m_doc.getElementsByTagName(elementTag);
        Element element = (Element) list.item(elementIndex);
        element.setAttribute(attributeTag, attributeValue);
    }

    public String renderXml() throws Exception {
        Transformer tf = TransformerFactory.newInstance().newTransformer();
        tf.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
        tf.setOutputProperty(OutputKeys.INDENT, "yes");
        Writer out = new StringWriter();
        tf.transform(new DOMSource(m_doc), new StreamResult(out));
        return new String(out.toString().getBytes(), StandardCharsets.UTF_8);
    }

    public static void main(String[] args) throws Exception {
        DOM_parser domParser = new DOM_parser("Popular_Baby_NAMES_NY.xml");
        FileWriter fileWriter = new FileWriter("dcc.xml");
        fileWriter.write(domParser.renderXml());
        fileWriter.close();
    }
}

```