УТВЕРЖДЕН

02068048. 42 5400.004.И2.04.1-1.М -ЛУ

**Lab5**

**Текст программы**

**02068048. 42 5400.004.И2.04.1-1.М**

**Листов 9**

2021

АННОТАЦИЯ

В данном программном документе приведен текст программы «lab5», предназначенной для преобразования формата данных. Исходным языком данной разработки является Java. Среда разработки IntelejIDEA.

Программа lab5 выполняет такие функции:

##### Преобразования формата данных с помощью файла rules

* Преобразования формата данных с помощью файла xml\_to\_xml.xsl

Оформление программного документа «Текст программы» произведено по требованиям ЕСПД (ГОСТ 19.101-77 [[1]](#footnote-1), ГОСТ 19.103-77 [[2]](#footnote-2), ГОСТ 19.104-78\* [[3]](#footnote-3), ГОСТ 19.105-78\* [[4]](#footnote-4), ГОСТ 19.106-78\* [[5]](#footnote-5), ГОСТ 19.401-78 [[6]](#footnote-6), ГОСТ 19.604-78\* [[7]](#footnote-7)).

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ. 3](#_Toc456977088)

[ОПИСАНИЕ ПРОГРАММЫ LAB5 НА ИСХОДНОМ ЯЗЫКЕ. 4](#_Toc456977088)

1. Описание программы LAB5 на исходном языке

Xml\_to\_xml.java

import org.w3c.dom.DOMException;

import org.w3c.dom.Document;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.\*;

import javax.xml.xpath.\*;

import java.net.\*;

import java.io.\*;

import java.util.HashMap;

import java.util.Set;

public class xmlToxml {

static Document document;

static int number\_doc = 0;

public static void main(String[] args) {

try {

DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();

document = documentBuilder.parse("source.xml");

TransformerFactory tFactory = TransformerFactory.newInstance();

Transformer transformer =

tFactory.newTransformer

(new javax.xml.transform.stream.StreamSource

("xml\_to\_xml.xsl"));

transformer.transform

(new javax.xml.transform.stream.StreamSource

("source.xml"),

new javax.xml.transform.stream.StreamResult

(new FileOutputStream("new\_xml.xml")));

} catch (Exception e) {

e.printStackTrace();

}

convert\_xml();

}

static HashMap<String, String> rules\_convert = new HashMap<>();

public static void convert\_xml() {

try {

File file = new File("rules");

FileReader fr = new FileReader(file);

BufferedReader reader = new BufferedReader(fr);

String line = reader.readLine();

while (line != null) {

String[] conv = line.split("'='");

rules\_convert.put(conv[0], conv[1]);

line = reader.readLine();

}

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

try {

File source = new File("source.xml");

File new\_source = new File("new\_source.xml");

FileReader fr = new FileReader(source);

BufferedReader reader = new BufferedReader(fr);

BufferedWriter writer = new BufferedWriter(new FileWriter(new\_source, true));

String line = reader.readLine();

while (line != null) {

int count = 0;

Set<String> keys = rules\_convert.keySet();

String lineSeparator = System.getProperty("line.separator");

for(String key : keys){

if(line.contains(key)){

//System.out.println(key+" "+line);

//printItem(document);

String newStr = rules\_convert.get(key);

String[] parametrs = newStr.split("'");

/\*

for(String s : parametrs){

System.out.print(s+" ");

}

System.out.println();

\*/

if(parametrs[0].equals("\"\"")) ;

else{

if(parametrs.length == 1){

writer.write(newStr+lineSeparator);

writer.write(lineSeparator);

writer.flush();

number\_doc++;

}

else{

String data = "";

for(int i = 1; i < parametrs.length;i+=2){

if(parametrs[i].contains(">")){

String[] bol\_f = parametrs[i].split(">");

String answer = printItem(document, bol\_f[0]);

if(Integer.parseInt(answer) > Integer.parseInt(bol\_f[1]))

data = "\""+"true"+"\"";

else data = "\""+"false"+"\"";

}

else if(parametrs[i].contains("<")){

String[] bol\_f = parametrs[i].split("<");

String answer = printItem(document, bol\_f[0]);

if(Integer.parseInt(answer) < Integer.parseInt(bol\_f[1]))

data = "\""+"true"+"\"";

else data = "\""+"false"+"\"";

}

else if(parametrs[i].contains("=")){

String[] bol\_f = parametrs[i].split("=");

String answer = printItem(document, bol\_f[0]);

if(Integer.parseInt(answer) == Integer.parseInt(bol\_f[1]))

data = "\""+"true"+"\"";

else data = "\""+"false"+"\"";

}

else data = "\""+printItem(document, parametrs[i])+"\"";

parametrs[i] = data;

data = "";

}

String wr = "";

for(String s : parametrs){

wr+=s;

}

writer.write(wr);

writer.flush();

writer.write(lineSeparator);

writer.flush();

}

}

count++;

}

}

if(count == 0) {

if(line.equals("</catalog>")) writer.write(line);

else writer.write(line+lineSeparator);

writer.flush();

writer.write(lineSeparator);

writer.flush();

}

line = reader.readLine();

}

writer.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (XPathExpressionException e) {

e.printStackTrace();

}

}

private static String printItem(Document document, String p) throws DOMException, XPathExpressionException {

XPathFactory pathFactory = XPathFactory.newInstance();

XPath xpath = pathFactory.newXPath();

// Пример записи XPath

// Подный путь до элемента

//XPathExpression expr = xpath.compile("BookCatalogue/Book/Cost");

// Все элементы с таким именем

//XPathExpression expr = xpath.compile("//Cost");

// Элементы, вложенные в другой элемент

XPathExpression expr = xpath.compile("//"+p);

NodeList nodes = (NodeList) expr.evaluate(document, XPathConstants.NODESET);

Node n = nodes.item(number\_doc);

return n.getTextContent();

}

}

}

Xml\_to\_xml.xsl

<?xml version="1.0"?>

<xsl:transform version="2.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

<xsl:output method="xml" encoding="UTF-8" indent="yes" />

<xsl:template match="catalog">

<catalog>

<xsl:apply-templates />

</catalog>

</xsl:template>

<xsl:template match="cd">

<cd>

<info>item="<xsl:value-of select="item"/>" artist="<xsl:value-of select="artist"/>" instock="<xsl:choose><xsl:when test="count > 0">true</xsl:when><xsl:otherwise>false</xsl:otherwise></xsl:choose>

</info>

<title><xsl:value-of select="artist" /> "<xsl:value-of select="title" />" - <xsl:value-of select="year"/></title>

<price><xsl:value-of select="price"/> <xsl:value-of select="currency"/></price>

</cd>

</xsl:template>

</xsl:transform>

rules

<cd>'='<cd item='item' artist='artist' instock='count>0'>

<item>'='""

<title>'='<title>'artist' 'title' - 'year'</title>

<artist>'='""

<country>'='""

<company>'='""

<price>'='<price>'price' 'currency'</price>

<currency>'='""

<year>'='""

<count>'='""

</cd>'='</cd>

Source.xml

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

<catalog>

<cd>

<item>53725</item>

<title>По ресторанам. Лучшие Хиты</title>

<artist>Набиев Руслан</artist>

<country>Россия</country>

<company>Zebra Studio</company>

<price>250</price>

<currency>руб</currency>

<year>2008</year>

<count>100</count>

</cd>

<cd>

<item>53726</item>

<title>Избранные Хиты</title>

<artist>Сектор Газа</artist>

<country>Россия</country>

<company>Showtime</company>

<price>250</price>

<currency>руб</currency>

<year>2021</year>

<count>0</count>

</cd>

</catalog>

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц) в докум. | № докумен-та | Входящий № сопроводит. докум. и дата | Подп. | Дата |
| Изм. | изменен-ных | заменен-ных | новых | аннули-рованных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

1. ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов [↑](#footnote-ref-1)
2. ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов [↑](#footnote-ref-2)
3. ГОСТ 19.104-78\* ЕСПД. Основные надписи [↑](#footnote-ref-3)
4. ГОСТ 19.105-78\* ЕСПД. Общие требования к программным документам [↑](#footnote-ref-4)
5. ГОСТ 19.106-78\* ЕСПД. Общие требования к программным документам, выполненным печатным способом [↑](#footnote-ref-5)
6. ГОСТ 19.401-78 ЕСПД. Текст программы. Требования к содержанию и оформлению [↑](#footnote-ref-6)
7. ГОСТ 19.604-78\* ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом [↑](#footnote-ref-7)