ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Факультет кибербезопасности и управления

Кафедра программного обеспечения и управления в технических системах

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №** **1**

по дисциплине Прикладное программирование

Xml.

|  |  |  |
| --- | --- | --- |
| **ВЫПОЛНИЛ** | | |
| студент | РПИС-11  (группа) | Ларкин М.Л.  (ФИО) |
|  | **ПРОВЕРИЛ** |  |
| Ассистент каф. ПрИ  (должность) | | Гильмуллин Р.И.  (ФИО) |

Самара 2023

**Задание на практическую работу**

В процессе написания тестовых заданий ознакомиться с элементами языка XML и средствами Java для работы с XML-документами.

**Задание 1**

Изучить предлагаемый файл описания типа документа сформировать документ согласно этим правилам.

--- group.dtd ----------------------------------------------------

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

**<!ELEMENT group (student\*)>**

**<!ELEMENT student (subject\*,average?)>**

**<!ATTLIST student**

**firstname CDATA #REQUIRED**

**lastname CDATA #REQUIRED**

**groupnumber CDATA #REQUIRED**

**>**

**<!ELEMENT subject EMPTY>**

**<!ATTLIST subject**

**title CDATA #REQUIRED**

**mark (1|2|3|4|5) #REQUIRED**

**>**

**<!ELEMENT average (#PCDATA)>**

------------------------------------------------------------------

Документ типа group.dtd описывает структуру XML-документа, который должен содержать элемент group, содержащий ноль или более элементов student. Каждый элемент student должен содержать ноль или более элементов subject и не более одного элемента average, а также атрибуты firstname, lastname и groupnumber. Каждый элемент subject должен содержать атрибуты title и mark, где mark может быть равен 1, 2, 3, 4 или 5. Элемент subject не может содержать других элементов. Элемент average может содержать только текстовое содержимое.

Для выполнения задания необходимо создать XML-документ, соответствующий описанию в файле group.dtd. В этом документе должны быть элементы group, student, subject и average, а также атрибуты firstname, lastname, groupnumber, title и mark. Каждый элемент должен быть расположен в соответствии с описанием в файле group.dtd.

**Пример XML-документа, соответствующего описанию в файле group.dtd:**

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

<!DOCTYPE group SYSTEM "group.dtd">

<group>

<student firstname="Maksim" lastname="Larkin" groupnumber="11">

<subject title="Math" mark="5"/>

<subject title="Physics" mark="4"/>

<average>4.5</average>

</student>

<student firstname="Ryan" lastname="Gosling" groupnumber="11">

<subject title="Math" mark="3"/>

<subject title="Physics" mark="4"/>

<average>3.5</average>

</student>

</group>

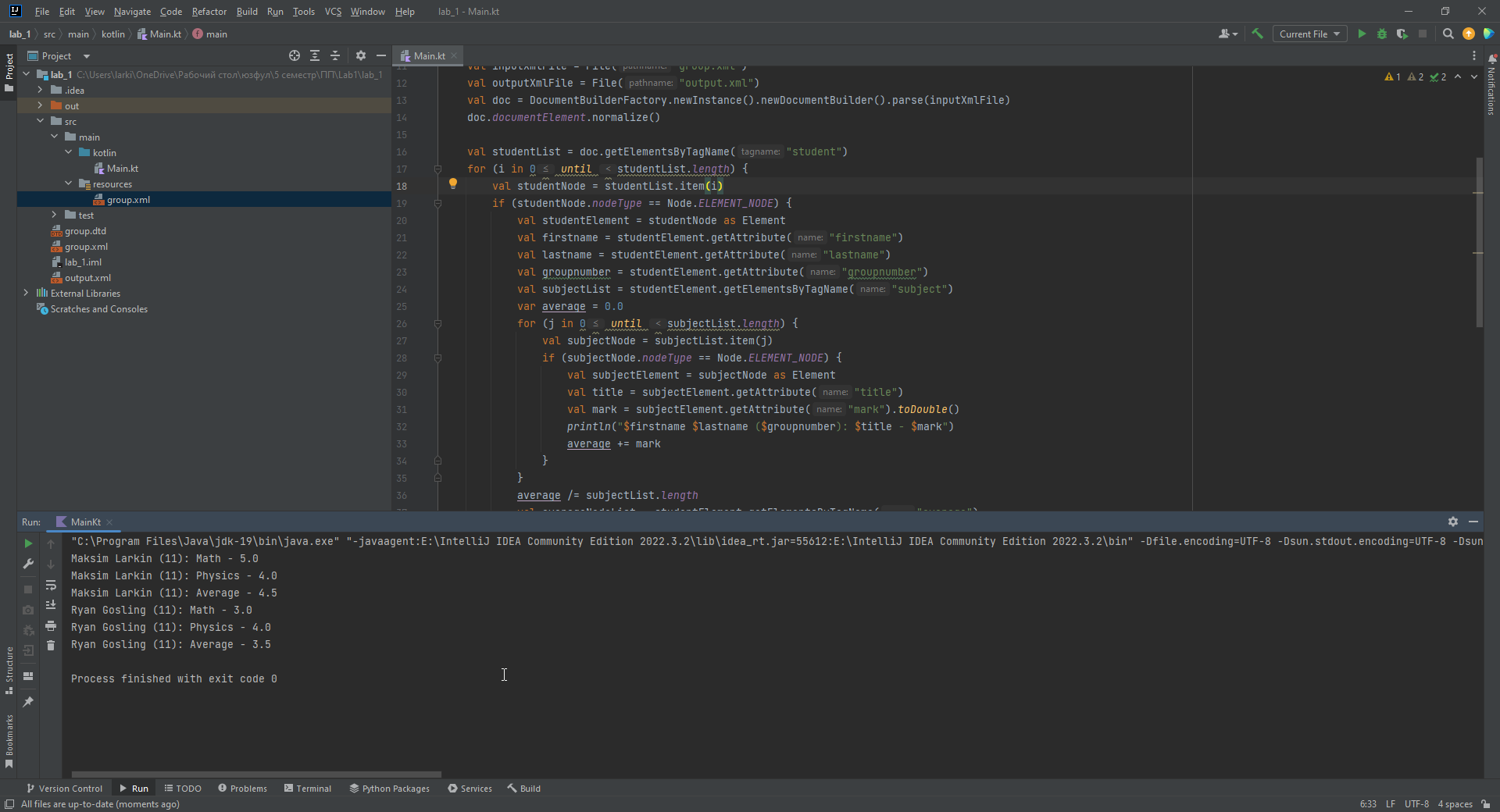
### Задание 2

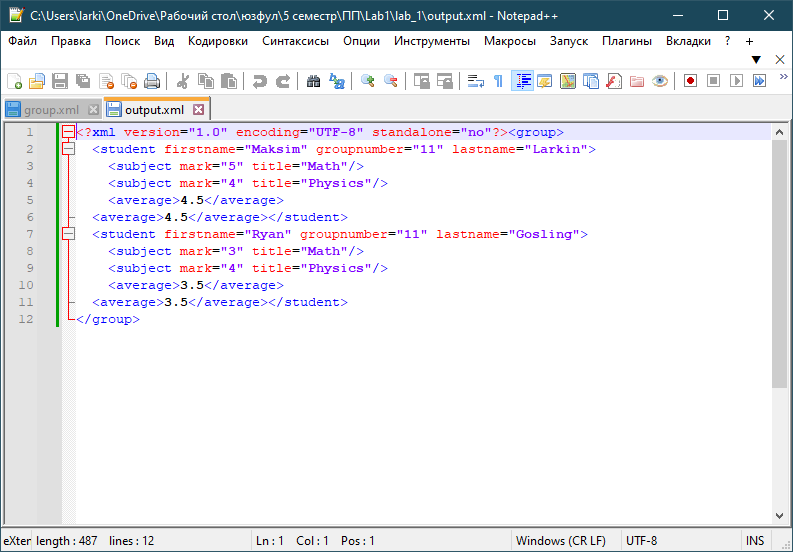
Разработать на Java консольное приложение, в котором задать имена входного и выходного XML-файлов. Задача приложения заключается в проверке значения средней оценки и его коррекции, если в исходном документе оно не соответствует действительности.

Для выполнения задания можно использовать SAX, DOM или JAXB.

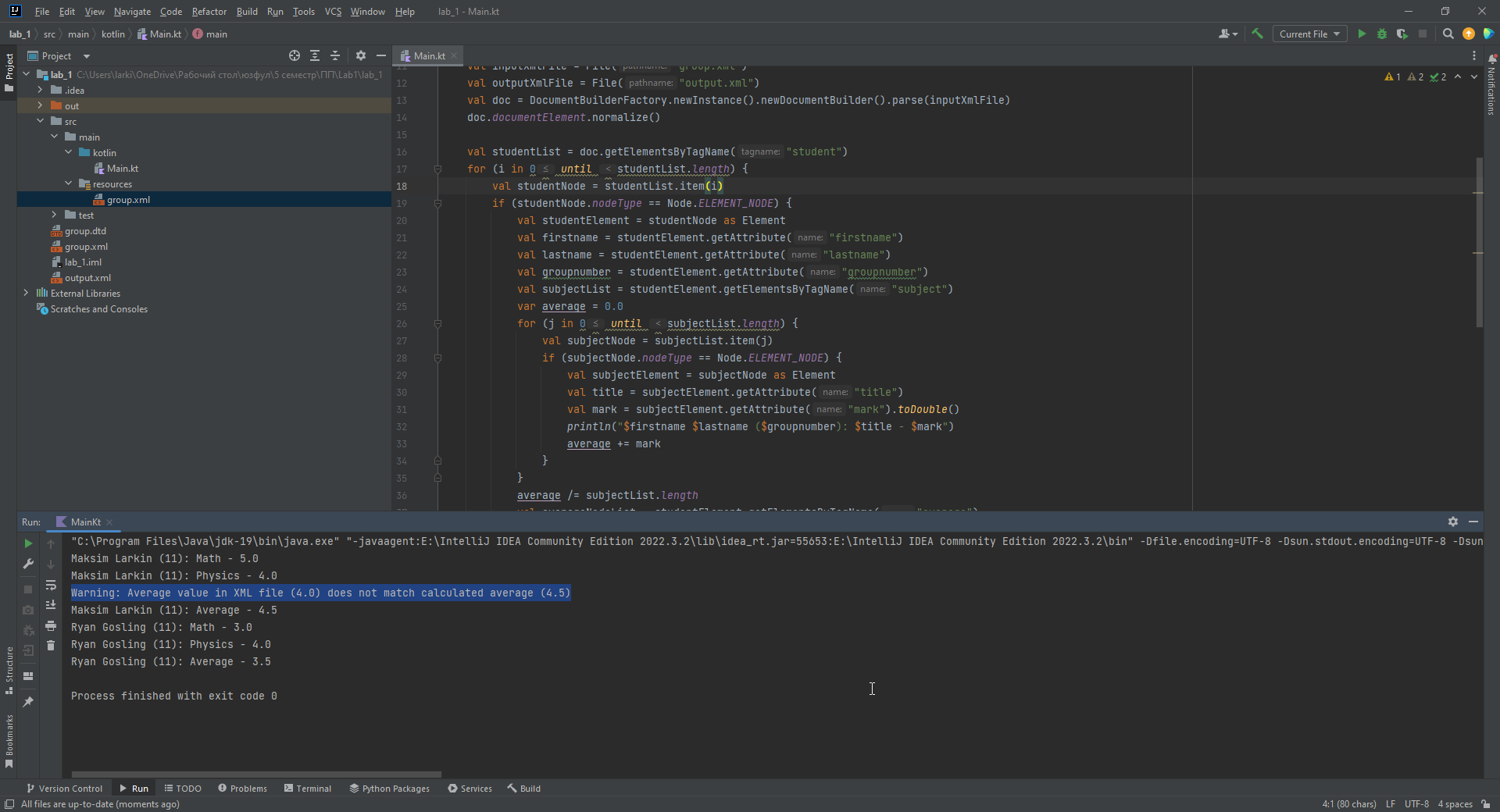
Сначала я создал второй xml файл (output.xml), для того, чтобы туда выводились проверенные средние оценки.

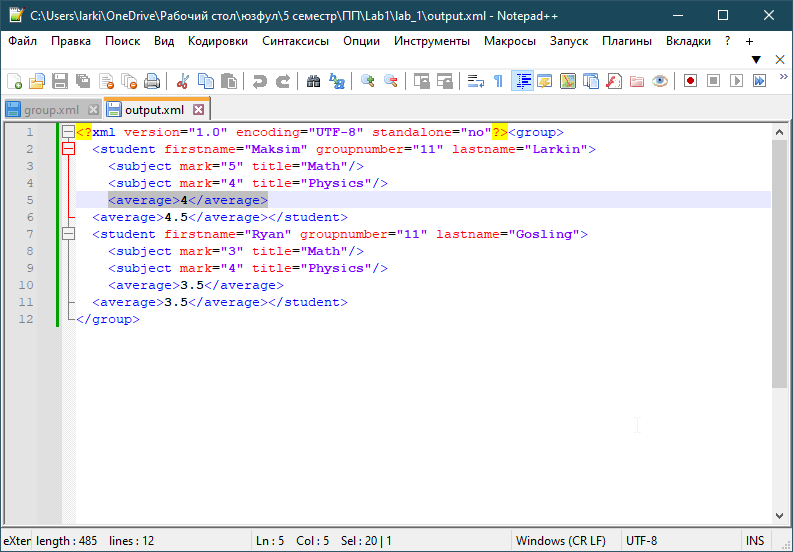
**Пример работы:**





Если в первом файле будет неправильный средний балл:





**Код программы:**

import org.w3c.dom.Element  
import org.w3c.dom.Node  
import org.w3c.dom.NodeList  
import java.io.File  
import javax.xml.parsers.DocumentBuilderFactory  
import javax.xml.transform.TransformerFactory  
import javax.xml.transform.dom.DOMSource  
import javax.xml.transform.stream.StreamResult  
  
fun main() {  
 val inputXmlFile = File("group.xml")  
 val outputXmlFile = File("output.xml")  
 val doc = DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(inputXmlFile)  
 doc.*documentElement*.normalize()  
  
 val studentList = doc.getElementsByTagName("student")  
 for (i in 0 *until* studentList.*length*) {  
 val studentNode = studentList.item(i)  
 if (studentNode.*nodeType* == Node.*ELEMENT\_NODE*) {  
 val studentElement = studentNode as Element  
 val firstname = studentElement.getAttribute("firstname")  
 val lastname = studentElement.getAttribute("lastname")  
 val groupnumber = studentElement.getAttribute("groupnumber")  
 val subjectList = studentElement.getElementsByTagName("subject")  
 var average = 0.0  
 for (j in 0 *until* subjectList.*length*) {  
 val subjectNode = subjectList.item(j)  
 if (subjectNode.*nodeType* == Node.*ELEMENT\_NODE*) {  
 val subjectElement = subjectNode as Element  
 val title = subjectElement.getAttribute("title")  
 val mark = subjectElement.getAttribute("mark").*toDouble*()  
 *println*("$firstname $lastname ($groupnumber): $title - $mark")  
 average += mark  
 }  
 }  
 average /= subjectList.*length* val averageNodeList = studentElement.getElementsByTagName("average")  
 if (averageNodeList.*length* > 0) {  
 val averageNode = averageNodeList.item(0)  
 if (averageNode.*nodeType* == Node.*ELEMENT\_NODE*) {  
 val averageElement = averageNode as Element  
 val averageValue = averageElement.*textContent*.*toDouble*()  
 if (average != averageValue) {  
 *println*("Warning: Average value in XML file ($averageValue) does not match calculated average ($average)")  
 }  
 }  
 } else {  
 *println*("Warning: No average value found in XML file")  
 }  
 *println*("$firstname $lastname ($groupnumber): Average - $average")  
 val newAverageElement = doc.createElement("average")  
 newAverageElement.*textContent* = average.toString()  
 studentElement.appendChild(newAverageElement)  
 }  
 }  
  
 val transformer = TransformerFactory.newInstance().newTransformer()  
 transformer.transform(DOMSource(doc), StreamResult(outputXmlFile))  
}

