Министерство образования Республики Беларусь

Учреждение образования

“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Факультет компьютерного проектирования

Дисциплина: СиТАиРИС

**ЛАБОРАТОРНАЯ РАБОТА № 4**

**«XML, работа с SAX и DOM парсерами»**

Выполнил:

Студент гр. 814301

Аблажевич В.В.

Проверил:

Лыщик А.П.

Минск 2020

**Задание**

Разработать программу и вывести на экран информацию о работниках.

Требуется:

• определить схему XML-файла;

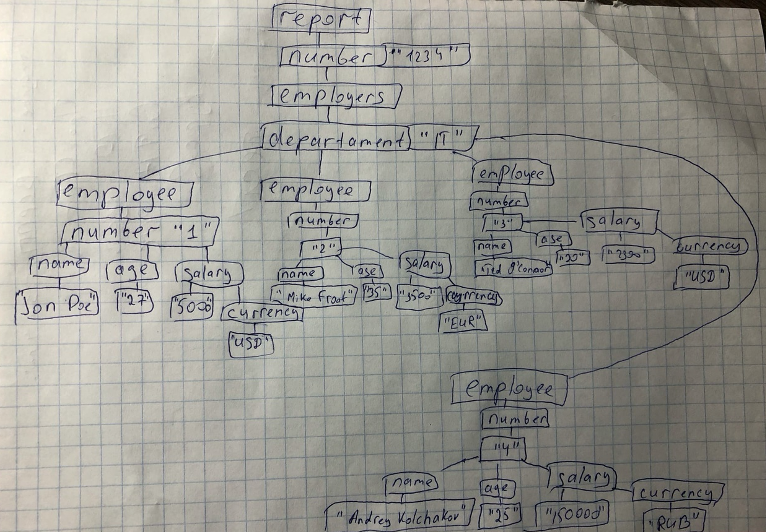
• определить структуры данных для представления в памяти содержимого XML-файла

после выполнения разбора (если такое представление необходимо по заданию);

• выбрать способ разбора (с создание документа или без такового);

• реализовать программу на языке программирования Java.

1. **Схема XML-файла**



1. **Текст программы**

В файле main мы работаем с SAX парсерами, инициализируем новые переменные для вызова из xml файла тегов с значениями.

**Main.java**

package main;

import java.io.File;

import java.io.IOException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.SAXException;

import java.util.List;

import java.util.ArrayList;

import org.w3c.dom.Node;

public class Main {

public static void main(String[] args) throws ParserConfigurationException, SAXException, IOException {

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

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

DocumentBuilder builder = factory.newDocumentBuilder();

Document document = builder.parse(file);

Element employersElement = (Element) document.getElementsByTagName("employers").item(0);

String departament = employersElement.getAttribute("departament");

NodeList employeeNodeList = document.getElementsByTagName("employee");

List<Employee> employeeList = new ArrayList<>();

for (int i=0; i < employeeNodeList.getLength(); i++){

if (employeeNodeList.item(i).getNodeType() == Node.ELEMENT\_NODE){

Element employeeElement = (Element) employeeNodeList.item(i);

Employee employee = new Employee();

employee.setDepartament(departament);

employee.setNumber(Integer.valueOf(employeeElement.getAttribute("number")));

NodeList childNodes = employeeElement.getChildNodes();

for(int j=0; j<childNodes.getLength(); j++){

if(childNodes.item(j).getNodeType() == Node.ELEMENT\_NODE){

Element childElement = (Element) childNodes.item(j);

switch (childElement.getNodeName()){

case "name": {

employee.setName(childElement.getTextContent());

} break;

case "age": {

employee.setAge(Integer.valueOf(childElement.getTextContent()));

} break;

case "salary" : {

employee.getSalary().setValue(Double.valueOf(childElement.getTextContent()));

employee.getSalary().setCurrency(childElement.getAttribute("currency"));

}

}

}

}

employeeList.add(employee);

}

}

employeeList.forEach(System.out::println);

}

}

class Employee{

private String departament;

private Integer number;

private String name;

private Integer age;

private Salary salary = new Salary();

public static class Salary{

private Double value;

private String currency;

public Double getValue(){

return value;

}

public void setValue(Double value){

this.value = value;

}

public String getCurrency(){

return currency;

}

public void setCurrency(String currency){

this.currency = currency;

}

}

public String getDepartament(){

return departament;

}

public void setDepartament(String departament){

this.departament = departament;

}

public Integer getNumber(){

return number;

}

public void setNumber(Integer number){

this.number = number;

}

public String getName(){

return name;

}

public void setName(String name){

this.name = name;

}

public Integer getAge(){

return age;

}

public void setAge(Integer age){

this.age = age;

}

public Salary getSalary(){

return salary;

}

@Override

public String toString(){

return "Employee(" +

"departament='" + departament + '\'' +

", number=" + number +

", name='" + name + '\'' +

", age=" + age +

", salary=" + salary.getValue() + " " + salary.getCurrency() +

')';

}

}

**File.xml**

В этом файле мы создаем основу программы, а именно древовидную структуру данных с помощью тегов и атрибутов.

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

<report number="1234">

<employers departament="IT">

<employee number ="1">

<name>John Doe</name>

<age>27</age>

<salary currency="USD">5000</salary>

</employee>

<employee number="2">

<name>Mike Froat</name>

<age>35</age>

<salary currency="EUR">3500</salary>

</employee>

<employee number="3">

<name>Ted O'Connor</name>

<age>20</age>

<salary currency="USD">2300</salary>

</employee>

<employee number="4">

<name>Andrey Kolchakov</name>

<age>25</age>

<salary currency="RUB">150000</salary>

</employee>

</employers>

</report>

1. **Результат выполнения программы**

Здесь мы видим результат выполнения нашей программы, а именно вывод информации о сотрудниках отдела с их отделом, именем, возрастом, номером, и зарплаты с названием валюты.

