**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 12**

по дисциплине «Визуальные средства разработки программных приложений»

на тему: «Обработка данных с использованием *XML*»

Выполнил: студент гр. ИТП-31

Сибирко В.Г.

Принял: преподаватель

Гуменников Е.Д.

Гомель 2021

**Цель работы:** изучить методы обработки данных с использованием *XML* в *Java*.

**Задание**

Выполнить задание, сохраняя объекты приложения в одном или нескольких файлах *XML*. Четные варианты используют *SAX*-парсер, нечетные *DOM*-парсер.

3. Система Больница. Пациенту назначается лечащий Врач. Врач может сделать назначение Пациенту (процедуры, лекарства, операции). Медсестра или другой Врач выполняет назначение. Пациент может быть выписан из Больницы по окончанию лечения, при нарушении режима или при иных обстоятельствах.

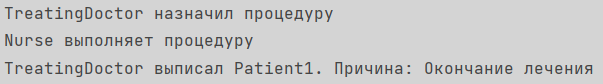


Рисунок 1 – Результат выполнения программы

Структура *xml* файла:

<list>

<hospital>

<doctors>

<doctor>

<doctorSurname>TreatingDoctor</doctorSurname>

</doctor>

<doctor>

<doctorSurname>Nurse</doctorSurname>

</doctor>

</doctors>

<patients>

<patient>

<patientSurname>Patient1</patientSurname>

<doctor>

<doctorSurname>Doctor1</doctorSurname>

</doctor>

<isHealthy>false</isHealthy>

</patient>

</patients>

</hospital>

</list>

**Вывод:** в результате лабораторной работы получены навыки по обработке данных с использованием *XML* в *Java*.

**Листинг программы:**

**Класс *Hospital.java***

package hospital;

import com.thoughtworks.xstream.annotations.XStreamAlias;

import java.io.Serializable;

import java.util.ArrayList;

@XStreamAlias("hospital")

public class Hospital implements Serializable {

@XStreamAlias("doctors")

private ArrayList<Doctor> doctors;

@XStreamAlias("patients")

private ArrayList<Patient> patients;

public Hospital(ArrayList<Doctor> doctors, ArrayList<Patient> patients) {

this.doctors = doctors;

this.patients = patients;

}

@Override

public String toString() {

return doctors.toString() + "\n" + patients.toString();

}

}

**Класс *Doctor.java***

package hospital;

import com.thoughtworks.xstream.annotations.XStreamAlias;

import java.io.Serializable;

@XStreamAlias("doctor")

public class Doctor implements Serializable {

@XStreamAlias("doctorSurname")

private String doctorSurname;

public Doctor(String doctorSurname) {

this.doctorSurname = doctorSurname;

}

public String destination(String destinationType) {

if (!destinationType.equals("")) {

System.out.println(doctorSurname + " назначил " + destinationType);

} else {

System.out.println(doctorSurname + " ничего не назначил");

}

return destinationType;

}

public void performingAssignment(String destinationType) {

if (!destinationType.equals("")) {

System.out.println(doctorSurname + " выполняет " + destinationType);

} else {

System.out.println("Врач ничего не назначал");

}

}

public void discharge(String patientSurname, String dischargeType) {

System.out.println(doctorSurname + " выписал " + patientSurname + ". Причина: " + dischargeType);

Patient.setIsHealthy(true);

}

@Override

public String toString() {

return doctorSurname;

}

}

**Класс *Patient.java***

package hospital;

import com.thoughtworks.xstream.annotations.XStreamAlias;

import java.io.Serializable;

@XStreamAlias("patient")

public class Patient implements Serializable {

@XStreamAlias("patientSurname")

private String patientSurname;

@XStreamAlias("doctor")

private Doctor doctor;

@XStreamAlias("isHealthy")

private boolean isHealthy = false;

public Patient(String patientSurname, Doctor doctor, boolean isHealthy) {

this.patientSurname = patientSurname;

this.doctor = doctor;

this.isHealthy = isHealthy;

}

@Override

public String toString() {

return "Фамилия: " + patientSurname +

"; Лечащий врач: " + doctor +

"; Состояние: " + isHealthy;

}

public String getPatientSurname() {

return patientSurname;

}

public static void setIsHealthy(boolean isHealthy) {

isHealthy = isHealthy;

}

}

**Класс *CustomSAXParser.java***

import hospital.\*;

import com.thoughtworks.xstream.XStream;

import com.thoughtworks.xstream.io.xml.DomDriver;

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

import javax.xml.parsers.SAXParser;

import javax.xml.parsers.SAXParserFactory;

import javax.xml.parsers.ParserConfigurationException;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.BufferedWriter;

import java.util.ArrayList;

public class CustomSAXParser {

private static final String HOSPITAL\_PATH = "..\\lab12\\hospital.xml";

private static final String PATH\_TO\_SAVE = "..\\lab12\\hospitalFromXMLHandler";

private static ArrayList<Hospital> hospitals = new ArrayList<>();

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

SAXParserFactory factory = SAXParserFactory.newInstance();

SAXParser parser = factory.newSAXParser();

XMLHandler handler = new XMLHandler();

parser.parse(new File(HOSPITAL\_PATH), handler);

for (Hospital hospital: hospitals) {

System.out.println(hospital);

}

ArrayList<Hospital> hospitalFromXMLHandler = hospitals;

marshaller(hospitalFromXMLHandler, PATH\_TO\_SAVE);

}

private static class XMLHandler extends DefaultHandler {

private ArrayList<Doctor> doctors;

private ArrayList<Patient> patients;

private String doctorSurname, patientSurname;

private boolean isHealthy;

private String thisElement;

@Override

public void startElement(String uri, String localName, String qName, Attributes attributes) {

thisElement = qName;

if (thisElement.equals("doctors")) {

doctors = new ArrayList<>();

}

if (thisElement.equals("patients")) {

patients = new ArrayList<>();

}

}

@Override

public void characters(char[] ch, int start, int length) {

String information = new String(ch, start, length);

information = information.replace("\n", "").trim();

if (!information.isEmpty()) {

switch (thisElement) {

case "doctorSurname" -> {

doctorSurname = information;

doctors.add(new Doctor(doctorSurname));

}

case "patientSurname" -> patientSurname = information;

case "isHealthy" -> {

isHealthy = Boolean.parseBoolean(information);

patients.add(new Patient(patientSurname, new Doctor(doctorSurname), isHealthy));

}

}

}

}

@Override

public void endElement(String uri, String localName, String qName) {

if (thisElement.equals("isHealthy")) {

hospitals.add(new Hospital(doctors, patients));

thisElement = "";

}

}

}

private static void saveToFile(String xml, String nameFile) throws IOException {

BufferedWriter writer = new BufferedWriter(new FileWriter(new File(nameFile + ".xml")));

writer.write(xml);

writer.close();

}

public static void marshaller(ArrayList<Hospital> hospitals, String pathToSave) throws IOException {

XStream xStream = new XStream(new DomDriver());

xStream.alias(pathToSave, ArrayList.class);

xStream.processAnnotations(Hospital.class);

String xml = xStream.toXML(hospitals);

saveToFile(xml, pathToSave);

}

}

**Класс *Main.java***

import hospital.\*;

import java.io.IOException;

import java.util.ArrayList;

public class Main {

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

final String PATH = "..\\lab12\\hospitalFromMain";

ArrayList<Hospital> hospitals = new ArrayList<>();

ArrayList<String> destinations = new ArrayList<>();

destinations.add("процедуру");

destinations.add("операцию");

destinations.add("выдачу лекарств на 2 дня");

ArrayList<Doctor> doctors = new ArrayList<>();

doctors.add(new Doctor("TreatingDoctor"));

doctors.add(new Doctor("Nurse"));

ArrayList<Patient> patients = new ArrayList<>();

patients.add(new Patient("Patient1", new Doctor("Doctor1"), false));

String destinationType = doctors.get(0).destination(destinations.get(0));

doctors.get(1).performingAssignment(destinationType);

doctors.get(0).discharge(patients.get(0).getPatientSurname(), "Окончание лечения");

hospitals.add(new Hospital(doctors, patients));

CustomSAXParser.marshaller(hospitals, PATH);

}

}