# Лабораторная работа № 5 студента группы ИТз-51 Черепченко Юрия Сергеевича Выполнение: \_\_\_\_\_\_\_\_\_\_\_ Защита: \_\_\_\_\_\_\_\_\_\_\_

# Хранение и обработка данных в Java.

**Цель работы:** изучение основ работы с файлами xml, properties и базами данных в java.

**Вариант-1**

# Задание к лабораторной работе

1. Разработать формат представления данных в xml и БД. В xml предусмотреть все рассмотренные виды узлов.
2. Реализовать методы: добавление, удаление и редактирование записей в каждом формате представления. Пользователю необходимо предоставить выбор формата представления, в котором будут производиться изменения.
3. Реализовать возможность конвертирования данных из xml в БД и из БД в xml.
4. Для чтения XML-документа использовать технологию SAX, для записи DOM.
5. Реализовать поиск по заданным параметрам. Поиск может осуществляться как по одному, так и по нескольким параметрам. Если параметр числовой, то должна быть возможность поиска максимальных и минимальных значений.
6. Минимальное количество записей в БД или xml 10.

# Ход работы

1. Ознакомился с теоретическими сведениями в методических указаниях и в дополнительном источнике.
2. Разработал программу, которая отвечает всем требованиям задания.

Текст программы

ConsoleMain

package ru.bstu.itz.cherepchenko.lab1;  
  
import java.util.Scanner;  
  
public class ConsoleMain {  
  
 private static String setValue() {  
 while (true) {  
 Scanner sc = new Scanner(System.*in*);  
 String str = sc.next();  
 if (str.length() > 0) {  
 return str;  
 } else {  
 System.*out*.println("Необходимо ввести значение!");  
 }  
 }  
 }  
  
 protected static String[] setValueDrug() {  
 String[] strings = new String[6];  
 System.*out*.println("Введите значения для лекарства:");  
 System.*out*.println("Введите название:");  
 strings[0] = *setValue*();  
 System.*out*.println("Введите тип:");  
 strings[1] = *setValue*();  
 System.*out*.println("Введите стоимость:");  
 strings[2] = *getNum*() + "";  
 System.*out*.println("Введите состав:");  
 strings[3] = *setValue*();  
 System.*out*.println("Введите противопоказания:");  
 strings[4] = *setValue*();  
 return strings;  
 }  
  
 private static Drug setNewDrug(int size) {  
 var strings = *setValueDrug*();  
 return new Drug(size + 1, strings[0], strings[1], Integer.*parseInt*(strings[2]),  
 strings[3], strings[4]);  
  
 }  
  
 private static void changeDrug(String filePath) {  
 var sax = new SAXParse();  
 var drugs = sax.readerSaxDocument(filePath);  
 System.*out*.println("Введите id лекарства:");  
 var searchId = *getNum*();  
 var strings = *setValueDrug*();  
 boolean flag = false;  
 for (int i = 0; i < drugs.size(); i++) {  
 if (drugs.get(i).getId() == searchId) {  
 drugs.set(i, new Drug(searchId,  
 strings[0], strings[1], Integer.*parseInt*(strings[2]),  
 strings[3], strings[4]));  
 flag = true;  
 break;  
 }  
 }  
 if (flag) {  
 var dom = new DomParse(filePath);  
 dom.setDomNodes(drugs);  
 } else {  
 System.*out*.println("Такого лекарства нет!");  
 }  
  
 }  
  
 private static void deleteDrug(String filePath) {  
 var sax = new SAXParse();  
 var drugs = sax.readerSaxDocument(filePath);  
 System.*out*.println("Введите id лекарства:");  
 var searchId = *getNum*();  
 boolean flag = false;  
 for (int i = 0; i < drugs.size(); i++) {  
 if (drugs.get(i).getId() == searchId) {  
 drugs.remove(i);  
 flag = true;  
 break;  
 }  
 }  
 if (flag) {  
 var dom = new DomParse(filePath);  
 dom.setDomNodes(drugs);  
 } else {  
 System.*out*.println("Такого лекарства нет!");  
 }  
  
 }  
  
 public static int getNum() {  
 Scanner sc = new Scanner(System.*in*);  
 while (!sc.hasNextInt()) {  
 System.*out*.println("Введенно некоректное значение!");  
 System.*out*.print("Введите значение повторно: ");  
 sc.next();  
 }  
 return sc.nextInt();  
 }  
  
 public static int choiceWithWork() {  
 System.*out*.println("Выберите с чем работать: \n" +  
 "1: XML\n" +  
 "2: БД\n" +  
 "3: Конвертировать данные из XML в БД\n" +  
 "4: Конвертировать данные из БД в XML\n" +  
 "5: Завершить работу.");  
 int type = 0;  
 while (true) {  
 type = *getNum*();  
 if (type >= 1 && type <= 5) {  
 return type;  
 }  
 System.*out*.println("Можно ввести только 1,2,3,4,5");  
 }  
 }  
  
 public static void start() {  
 var prop = new PropertiesParse();  
 var catalog = prop.readCatalogRoot();  
 String filePath = catalog + "file.xml";  
 int type = *choiceWithWork*();  
 if (type == 3) {  
 var sax = new SAXParse();  
 var parsing = new Parsing(sax.readerSaxDocument(filePath));  
 parsing.parseXMLtoDB();  
 *start*();  
 }  
 if (type == 4) {  
 var sax = new SAXParse();  
 var dom = new DomParse(filePath);  
 var parsing = new Parsing(sax.readerSaxDocument(filePath), dom);  
 parsing.parseDBtoXML();  
 *start*();  
 }  
 if (type == 5) return;  
 while (true) {  
 System.*out*.println("Выберите действие: \n" +  
 "1: Вывести всё содержимое\n" +  
 "2: Найти содержимое по параметру\n" +  
 "3: Добавить новую запись\n" +  
 "4: Изменить запись\n" +  
 "5: Удалить запись\n" +  
 "9: Выбрать заново с чем работать\n" +  
 "0: Завершить работу\n");  
 var choice = *getNum*();  
 switch (choice) {  
 case 1: {  
 if (type == 1) {  
 var sax = new SAXParse();  
 var drugs = sax.readerSaxDocument(filePath);  
 if (drugs.size() > 0) {  
 for (Drug drug : drugs) {  
 System.*out*.println(drug.toString());  
 }  
 }  
 } else if (type == 2) {  
 var mySqlObj = new MySqlParse();  
 var result = mySqlObj.workDataBase(choice);  
 try {  
 while (result.next()) {  
 Drug drug = new Drug(  
 result.getInt("id"),  
 result.getString("name"),  
 result.getString("type"),  
 result.getInt("cost"),  
 result.getString("structure"),  
 result.getString("contraindications")  
 );  
 System.*out*.println(drug);  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 break;  
 case 2: {  
 if (type == 1) {  
 var sax = new SAXParse();  
 String content = "";  
 System.*out*.print("Выберите содержимое поиска (id):\n");  
 Scanner scanner = new Scanner(System.*in*);  
 content = scanner.nextLine();  
 var drug = sax.searchSaxDocument(filePath, content);  
 System.*out*.println(drug != null ? drug.toString() : "Такого лекарства нет!");  
 } else if (type == 2) {  
 var mySqlObj = new MySqlParse();  
 var result = mySqlObj.workDataBase(choice);  
 try {  
 while (result.next()) {  
 Drug drug = new Drug(  
 result.getInt("id"),  
 result.getString("name"),  
 result.getString("type"),  
 result.getInt("cost"),  
 result.getString("structure"),  
 result.getString("contraindications")  
 );  
 System.*out*.println(drug.toString());  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 break;  
 case 3: {  
 if (type == 1) {  
 var sax = new SAXParse();  
 var drugs = sax.readerSaxDocument(filePath);  
 var newDrug = *setNewDrug*(drugs.size());  
 drugs.add(newDrug);  
 var dom = new DomParse(filePath);  
 dom.setDomNodes(drugs);  
 } else if (type == 2) {  
 var mySqlObj = new MySqlParse();  
 mySqlObj.workDataBase(choice);  
 }  
 }  
 break;  
 case 4: {  
 if (type == 1) {  
 *changeDrug*(filePath);  
 } else if (type == 2) {  
 var mySqlObj = new MySqlParse();  
 mySqlObj.workDataBase(choice);  
 }  
 }  
 break;  
 case 5: {  
 if (type == 1) {  
 *deleteDrug*(filePath);  
 } else if (type == 2) {  
 var mySqlObj = new MySqlParse();  
 mySqlObj.workDataBase(choice);  
 }  
 }  
 break;  
 case 9: {  
 *start*();  
 }  
 break;  
 case 0: {  
 return;  
 }  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 *start*();  
 }  
}

Drug

package ru.bstu.itz.cherepchenko.lab1;  
  
public class Drug {  
 protected int id;  
 protected String name;  
 protected String type;  
 protected int cost;  
 protected String structure;  
 protected String contraindications;  
  
 public Drug(int id, String name, String type, int cost, String structure, String contraindications) {  
 this.id = id;  
 this.name = name;  
 this.type = type;  
 this.cost = cost;  
 this.structure = structure;  
 this.contraindications = contraindications;  
 }  
 @Override  
 public String toString() {  
 return "Id:"+this.id + "\n"  
 +"Название: " + this.name + "\n"  
 +"Тип: " + this.type + "\n"  
 +"Цена: " + this.cost + "\n"  
 +"Состав: " + this.structure + "\n"  
 +"Противопоказания: " + this.contraindications + "\n";  
 }  
  
 public String getStudent() {  
 return this.toString();  
 }  
  
 public int getId() {  
 return this.id;  
 }  
  
 public String getName() {  
 return this.name;  
 }  
 public String getType() {  
 return this.type;  
 }  
 public int getCost() {  
 return this.cost;  
 }  
 public String getStructure() {  
 return this.structure;  
 }  
 public String getContraindications() {  
 return this.contraindications;  
 }  
}

DomParse

package ru.bstu.itz.cherepchenko.lab1;  
  
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.File;  
import java.util.ArrayList;  
  
public class DomParse {  
 private final String filePath;  
 DomParse(String filepath) {  
 this.filePath = filepath;  
 }  
  
 public void getDOMNodes() {  
 File xmlFile = new File(this.filePath);  
 DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();  
 DocumentBuilder builder;  
 try {  
 builder = factory.newDocumentBuilder();  
 Document document = builder.parse(xmlFile);  
 // теперь XML полностью загружен в память  
 //вернули корневой элемент документа и нормализовали  
 document.getDocumentElement().normalize();  
 System.*out*.println("Корневой элемент: " + document.getDocumentElement().getNodeName());  
 // получаем узлы с именем Language  
 NodeList nodeList = document.getElementsByTagName("Language");  
 for (int i = 0; i < nodeList.getLength(); i++) {  
 Node node = nodeList.item(i);  
 // если узел является элементом  
 if (node.getNodeType() == Node.*ELEMENT\_NODE*) {  
 Element element = (Element) node;  
 System.*out*.println("\nЭлемент "+element.getNodeName());  
  
 // получаем список дочерних  
 NodeList childNodesList = element.getChildNodes();  
 for (int j = 0; j < childNodesList.getLength(); j++) {  
 Node childNode = childNodesList.item(j);  
 // если узел является элементом  
 if (childNode.getNodeType() == Node.*ELEMENT\_NODE*) {  
 System.*out*.println(childNode.getNodeName()+": "+ childNode.getTextContent());  
 }  
 }  
 }  
 }  
 } catch (Exception exc) {  
 exc.printStackTrace();  
 }  
 }  
  
 public void setDomNodes(ArrayList<Drug> drugs) {  
 DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();  
 DocumentBuilder builder;  
 try {  
 builder = factory.newDocumentBuilder();  
 Document doc = builder.newDocument();// создаем пустой объект Document  
 // создаем корневой элемент  
 Element rootElement = doc.createElement("Drugs");  
 doc.appendChild(rootElement);  
 // добавляем корневой элемент в объект Document  
 for(int i = 0; i < drugs.size(); i++){  
 rootElement.appendChild(*getDrug*(doc, i + 1, drugs.get(i).getName(), drugs.get(i).getType(),  
 drugs.get(i).getCost(), drugs.get(i).getStructure(), drugs.get(i).getContraindications()));  
 }  
 doc.getDocumentElement().normalize();  
 //создаем объект TransformerFactory для преобразования документа в файл  
 TransformerFactory transformerFactory = TransformerFactory.*newInstance*();  
 Transformer transformer = transformerFactory.newTransformer();  
 // установка параметров форматирования для красивого вывода  
 transformer.setOutputProperty(OutputKeys.*INDENT*, "yes");  
 transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");  
 //получение исходного кода готового документа  
 DOMSource source = new DOMSource(doc);  
 //создание объекта для записи - файл  
 StreamResult file = new StreamResult(new File(this.filePath));  
 //запись данных  
 transformer.transform(source, file);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 // метод для создания нового узла XML-файла  
 private static Node getDrug(Document doc, int id, String name, String type, int cost, String structure, String contraindications) {  
 Element language = doc.createElement("Drug");  
 language.setAttribute("id", id+""); // устанавливаем атрибут id  
 // создаем элементы name и age  
 language.appendChild(*getLanguageElements*(doc, "name", name));  
 language.appendChild(*getLanguageElements*(doc, "type", type));  
 language.appendChild(*getLanguageElements*(doc, "cost", cost + ""));  
 language.appendChild(*getLanguageElements*(doc, "structure", structure));  
 language.appendChild(*getLanguageElements*(doc, "contraindications", contraindications));  
 return language;  
 }  
  
 // метод для создания одного узла  
 private static Node getLanguageElements(Document doc, String name, String value) {  
 Element node = doc.createElement(name);  
 node.appendChild(doc.createTextNode(value));  
 return node;  
 }  
  
}

MySqlParse

package ru.bstu.itz.cherepchenko.lab1;  
  
import java.sql.\*;  
import java.util.Scanner;  
  
public class MySqlParse {  
 private String connectionUrl;  
 private String userName ;  
 private String password ;  
 private Statement statement;  
  
 private static String setValue() {  
 while (true) {  
 Scanner sc = new Scanner(System.*in*);  
 String str = sc.next();  
 if (str.length() > 0) {  
 return str;  
 } else {  
 System.*out*.println("Необходимо ввести значение!");  
 }  
 }  
 }  
  
 protected static String[] setValueDrug() {  
 String[] strings = new String[6];  
 System.*out*.println("Введите значения для лекарства:");  
 System.*out*.println("Введите название:");  
 strings[0] = *setValue*();  
 System.*out*.println("Введите тип:");  
 strings[1] = *setValue*();  
 System.*out*.println("Введите стоимость:");  
 strings[2] = *getNum*() + "";  
 System.*out*.println("Введите состав:");  
 strings[3] = *setValue*();  
 System.*out*.println("Введите противопоказания:");  
 strings[4] = *setValue*();  
 return strings;  
 }  
  
 public static int getNum() {  
 Scanner sc = new Scanner(System.*in*);  
 while (!sc.hasNextInt()) {  
 System.*out*.println("Введенно некоректное значение!");  
 System.*out*.print("Введите значение повторно: ");  
 sc.next();  
 }  
 return sc.nextInt();  
 }  
 MySqlParse() {  
 try {  
 PropertiesParse propertiesParse = new PropertiesParse();  
 var settings = propertiesParse.bdSettings();  
 this.connectionUrl = settings[0];  
 this.userName = settings[1];  
 this.password = settings[2];  
 Connection connection = DriverManager.*getConnection*(connectionUrl, userName, password);  
 System.*out*.println("Подключение прошло успешно!");  
 this.statement = (Statement) connection.createStatement();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 public ResultSet getAll() throws SQLException {  
 return this.statement.executeQuery("select \* from drugs");  
 }  
  
 public ResultSet searchRecord(int id) throws SQLException {  
 return statement.executeQuery("select \* from drugs where id in(" + id + ");");  
 }  
  
 public void addNewRecord(String[] strings) throws SQLException {  
 this.statement.executeUpdate("INSERT INTO drugs (name, type, cost, structure, contraindications)" +  
 " VALUES ('" + strings[0] + "','" + strings[1] + "', " + strings[2] + ", '" + strings[3] + "', " +  
 "'" + strings[4] + "')");  
 }  
  
 public void addNewRecord(Drug drug) throws SQLException {  
 statement.executeUpdate("INSERT INTO drugs (name, type, cost, structure, contraindications)" +  
 " VALUES ('" + drug.name + "','" + drug.type + "', " + drug.cost + ", '" + drug.structure + "', " +  
 "'" + drug.contraindications + "')");  
 }  
  
 public void updateRecord(int id, Drug drug) throws SQLException {  
 if (drug != null) {  
 statement.executeUpdate("update drugs set name = '" + drug.getName() + "', type = '" + drug.getType() + "', cost = " + drug.getCost() + ", structure = '" + drug.getStructure() + "', contraindications = '" + drug.getContraindications() + "' where id = " + id + ";");  
 } else {  
 var strings = *setValueDrug*();  
 statement.executeUpdate("update drugs set name = '" + strings[0] + "', type = '" + strings[1] + "', cost = " + strings[2] + ", structure = '" + strings[3] + "', contraindications = '" + strings[4] + "' where id = " + id + ";");  
 }  
  
  
 }  
  
 public void deleteRecord(int id) throws SQLException {  
 statement.executeUpdate("delete from drugs where id in(" + id + ");");  
 }  
  
  
 public ResultSet workDataBase(int action) {  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 try {  
 switch (action) {  
 case 1 -> {  
 return getAll();  
 }  
 case 2 -> {  
 System.*out*.println("Введите Id записи");  
 var id = *getNum*();  
 return searchRecord(id);  
 }  
 case 3 -> {  
 var strings = *setValueDrug*();  
 addNewRecord(strings);  
 }  
 case 4 -> {  
 System.*out*.println("Введите Id записи");  
 var id = *getNum*();  
 updateRecord(id, null);  
 }  
 case 5 -> {  
 System.*out*.println("Введите Id записи");  
 var id = *getNum*();  
 deleteRecord(id);  
 }  
 }  
 } catch (Exception e) {  
 var prop = new PropertiesParse();  
 System.*out*.println(prop.error());  
 }  
 } catch (Exception e) {  
 var prop = new PropertiesParse();  
 System.*out*.println(prop.errorDriver());  
 }  
 return null;  
 }  
  
}

Parsing

package ru.bstu.itz.cherepchenko.lab1;  
  
import java.util.ArrayList;  
  
public class Parsing {  
 protected ArrayList<Drug> drugs = new ArrayList<Drug>();  
 protected DomParse domParse;  
  
 public Parsing(ArrayList<Drug> studentsXML) {  
 this.drugs = studentsXML;  
 }  
  
 public Parsing(ArrayList<Drug> studentsXML, DomParse domParse) {  
 this.drugs = studentsXML;  
 this.domParse = domParse;  
 }  
  
 public void parseXMLtoDB() {  
 var mySqlObj = new MySqlParse();  
 for (Drug drug : this.drugs) {  
 try {  
 mySqlObj.addNewRecord(drug);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 public void parseDBtoXML() {  
 var mySqlObj = new MySqlParse();  
 try {  
 var result = mySqlObj.getAll();  
 while (result.next()) {  
 this.drugs.add(new Drug(  
 result.getInt("id"),  
 result.getString("name"),  
 result.getString("type"),  
 result.getInt("cost"),  
 result.getString("structure"),  
 result.getString("contraindications")  
 ));  
 }  
 this.domParse.setDomNodes(this.drugs);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

PropertiesParse

package ru.bstu.itz.cherepchenko.lab1;  
  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.util.Properties;  
  
public class PropertiesParse {  
 private final String propPath = "settings.properties";  
 PropertiesParse() {  
  
 }  
 public String readCatalogRoot () {  
 Properties prop = new Properties();  
 try {  
 FileInputStream fis = new FileInputStream(propPath);  
 prop.load(fis);  
 return new String(prop.getProperty("rootCatalog").getBytes("ISO8859-1"));  
 } catch (IOException e) {  
 System.*out*.println("Ошибка в программе: файл не найден");  
 e.printStackTrace();  
 }  
 return "";  
 }  
 public String[] bdSettings() {  
 Properties prop = new Properties();  
 try {  
 //обращение к файлу и получение данных  
 FileInputStream fis = new FileInputStream(propPath);  
 prop.load(fis);  
 // взятие свойства и преобразование в необходимую кодировку  
 String uri = new String(prop.getProperty("connectionUrl").getBytes("ISO8859-1"));  
 String name = new String(prop.getProperty("userName").getBytes("ISO8859-1"));  
 String password = new String(prop.getProperty("password").getBytes("ISO8859-1"));  
 String[] propStr = {uri,name,password};  
 return propStr;  
  
 } catch (IOException e) {  
 System.*out*.println("Ошибка в программе: файл не найден");  
 e.printStackTrace();  
 }  
 String[] propStr = {};  
 return propStr;  
 }  
 public String errorDriver() {  
 Properties prop = new Properties();  
 try {  
 FileInputStream fis = new FileInputStream(propPath);  
 prop.load(fis);  
 return new String(prop.getProperty("errorDriver").getBytes("ISO8859-1"));  
 } catch (IOException e) {  
 System.*out*.println("Ошибка в программе: файл не найден");  
 e.printStackTrace();  
 }  
 return "";  
 }  
  
 public String error() {  
 Properties prop = new Properties();  
 try {  
 FileInputStream fis = new FileInputStream(propPath);  
 prop.load(fis);  
 return new String(prop.getProperty("error").getBytes("ISO8859-1"));  
 } catch (IOException e) {  
 System.*out*.println("Ошибка в программе: файл не найден");  
 e.printStackTrace();  
 }  
 return "";  
 }  
}

SAXParse

package ru.bstu.itz.cherepchenko.lab1;  
  
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.stream.XMLOutputFactory;  
import javax.xml.stream.XMLStreamWriter;  
import java.io.File;  
import java.io.FileOutputStream;  
import java.util.ArrayList;  
import java.util.Arrays;  
  
public class SAXParse {  
 protected String content;  
 protected boolean flag = false;  
 protected ArrayList<Drug> drugs = new ArrayList<Drug>();  
 protected String[] strMas = new String[6];  
 protected Drug drug;  
 protected int step = 0;  
 private final DefaultHandler handler = new DefaultHandler() {  
 String tag = "";  
 String id = "";  
  
 @Override  
 public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {  
 tag = qName;  
 if (tag.equalsIgnoreCase("Drug"))  
 id = attributes.getValue("id");  
 }  
  
 @Override  
 public void characters(char ch[], int start, int length) throws SAXException {  
 if (tag.equalsIgnoreCase("name")) {  
 strMas[0] = new String(ch, start, length);  
 } else if (tag.equalsIgnoreCase("type")) {  
 strMas[1] = new String(ch, start, length);  
 } else if (tag.equalsIgnoreCase("cost")) {  
 strMas[2] = new String(ch, start, length);  
 } else if (tag.equalsIgnoreCase("structure")) {  
 strMas[3] = new String(ch, start, length);  
 } else if (tag.equalsIgnoreCase("contraindications")) {  
 strMas[4] = new String(ch, start, length);  
 }  
  
 if (strMas[0] != null && strMas[1] != null && strMas[2] != null &&  
 strMas[3] != null && strMas[4] != null && !id.equals("")) {  
 Drug drug = new Drug(Integer.*parseInt*(id), strMas[0], strMas[1], Integer.*parseInt*(strMas[2]), strMas[3], strMas[4]);  
 Arrays.*fill*(strMas, null);  
 drugs.add(drug);  
 }  
 }  
  
 @Override  
 public void endElement(String uri, String localName, String qName) throws SAXException {  
 tag = "";  
 }  
 };  
  
 private final DefaultHandler handlerSearch = new DefaultHandler() {  
 String tag = "";  
 String id = "";  
  
 @Override  
 public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {  
 tag = qName;  
 if (tag.equalsIgnoreCase("Drug"))  
 id = attributes.getValue("id");  
 }  
  
 @Override  
 public void characters(char ch[], int start, int length) throws SAXException {  
 if (id.equalsIgnoreCase(content)) {  
 if (tag.equalsIgnoreCase("name") && strMas[0] == null) {  
 strMas[0] = new String(ch, start, length);  
 } else if (tag.equalsIgnoreCase("type") && strMas[1] == null) {  
 strMas[1] = new String(ch, start, length);  
 } else if (tag.equalsIgnoreCase("cost") && strMas[2] == null) {  
 strMas[2] = new String(ch, start, length);  
 } else if (tag.equalsIgnoreCase("structure") && strMas[3] == null) {  
 strMas[3] = new String(ch, start, length);  
 } else if (tag.equalsIgnoreCase("contraindications") && strMas[4] == null) {  
 strMas[4] = new String(ch, start, length);  
 }  
 if (strMas[0] != null && strMas[1] != null && strMas[2] != null &&  
 strMas[3] != null && strMas[4] != null && !id.equals("")) {  
 drug = new Drug(Integer.*parseInt*(id), strMas[0], strMas[1], Integer.*parseInt*(strMas[2]), strMas[3], strMas[4]);  
 }  
 }  
 }  
  
 @Override  
 public void endElement(String uri, String localName, String qName) throws SAXException {  
 if (tag.equalsIgnoreCase(""))  
 flag = false;  
 if (tag.equalsIgnoreCase("name") ||  
 tag.equalsIgnoreCase("type") ||  
 tag.equalsIgnoreCase("cost") ||  
 tag.equalsIgnoreCase("structure") ||  
 tag.equalsIgnoreCase("contraindications")  
 )  
 tag = "";  
 }  
 };  
  
 public Drug searchSaxDocument(String filePath, String content) {  
 try {  
 SAXParserFactory factory = SAXParserFactory.*newInstance*();  
 SAXParser saxParser = factory.newSAXParser();  
 this.content = content;  
 saxParser.parse(new File(filePath), handlerSearch);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return drug;  
 }  
  
 public ArrayList<Drug> readerSaxDocument(String filePath) {  
 try {  
 SAXParserFactory factory = SAXParserFactory.*newInstance*();  
 SAXParser saxParser = factory.newSAXParser();  
 saxParser.parse(new File(filePath), this.handler);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 return this.drugs;  
 }  
  
 //В качестве аргумента принимает путь файла, в который нужно записать  
 public void writeSaxDocument(String filePath) {  
 try {  
 XMLOutputFactory xmlOutputFactory = XMLOutputFactory.*newFactory*();  
 XMLStreamWriter writer = xmlOutputFactory.createXMLStreamWriter(new FileOutputStream(filePath));  
 writer.writeStartDocument();  
 writer.writeCharacters("\n");  
 writer.writeStartElement("root");  
 writer.writeCharacters("\n");  
 writer.writeStartElement("font");  
 writer.writeAttribute("id", "1");  
 writer.writeCharacters("TimesNewRoman\n");  
 writer.writeEndElement();  
 writer.writeEndElement();  
 writer.writeEndDocument();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void searchSaxDocument(String filePath, int searchId) {  
 }  
};

Таблица:   
create table drugs  
(  
 name varchar(512) null,  
 type varchar(256) null,  
 cost int null,  
 structure varchar(512) null,  
 contraindications varchar(512) null,  
 id bigint unsigned auto\_increment,  
 constraint id  
 unique (id)  
);

Вывод: изучил основы работы с файлами xml, properties и базами данных в java.