Парсинг Dom

package com.company;  
  
import org.w3c.dom.Document;  
import org.w3c.dom.NamedNodeMap;  
import org.w3c.dom.Node;  
import org.w3c.dom.NodeList;  
import org.xml.sax.SAXException;  
  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import java.io.File;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.HashMap;  
  
public class Main {  
 public static class Apartment {  
 String sellerName;  
 String sellerPhoneNumber;  
 String regionName;  
 Double square;  
 Double kitchenSquare;  
 Double housingPartSquare;  
 ArrayList<Double> roomsSquare = new ArrayList<Double>();  
  
 public Apartment(String sellerName, String sellerPhoneNumber, String regionName,  
 Double square, Double kitchenSquare, Double housingPartSquare,  
 ArrayList<Double> roomsSquare) {  
 this.sellerName = sellerName;  
 this.sellerPhoneNumber = sellerPhoneNumber;  
 this.regionName = regionName;  
 this.kitchenSquare = kitchenSquare;  
 this.square = square;  
 this.housingPartSquare = housingPartSquare;  
 this.roomsSquare = roomsSquare;  
 }  
  
 @Override  
 public final String toString() {  
 String \_roomsSquare = "";  
 for (var square : roomsSquare) {  
 \_roomsSquare += square + " ";  
 }  
 return " sellerName = " + sellerName + " sellerPhoneNumber = " + sellerPhoneNumber + " regionName = " + regionName +  
 " square = " + square.toString() + " kitchenSquare = " + kitchenSquare.toString() + " housingPartSquare = " + housingPartSquare.toString() +  
 " roomsSquare = " + \_roomsSquare;  
 }  
 }  
  
 private static void printInfoAboutAllChildNodes(NodeList list) {  
 for (int i = 0; i < list.getLength(); i++) {  
 Node node = list.item(i);  
 if (node.getNodeType() == Node.*TEXT\_NODE*) {  
 String textInformation = node.getNodeValue().replace("\n", "").trim();  
 if (!textInformation.isEmpty())  
 System.*out*.println("Внутри элемента найден текст: " + node.getNodeValue());  
 } else {  
 System.*out*.println("Найден элемент: " + node.getNodeName() + ", его атрибуты:");  
 NamedNodeMap attributes = node.getAttributes();  
 for (int k = 0; k < attributes.getLength(); k++)  
 System.*out*.println("Имя атрибута: " + attributes.item(k).getNodeName() + ", его значение: " + attributes.item(k).getNodeValue());  
 }  
 if (node.hasChildNodes())  
 *printInfoAboutAllChildNodes*(node.getChildNodes());  
 }  
 }  
  
 public static void main(String[] args) throws ParserConfigurationException, IOException, SAXException {  
 File fXmlFile = new File("info.xml");  
 DocumentBuilderFactory dbFactory = DocumentBuilderFactory.*newInstance*();  
 DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();  
 Document document = dBuilder.parse(fXmlFile);  
 //NodeList employeeElements = document.getDocumentElement().getElementsByTagName("flat");  
 NodeList matchedElementsList = document.getElementsByTagName("flat");  
 Node flat = matchedElementsList.item(0);  
 String region = flat.getParentNode().getAttributes().item(0).getNodeValue();  
 NodeList flatChilds = flat.getChildNodes();  
 String sellerName = "";  
 String sellerPhone = "";  
 Double square;  
 Double kitchenSquare;  
 Double housingPartSquare;  
 for (int i = 0; i < flatChilds.getLength(); i++) {  
 if (flatChilds.item(i).getNodeName() == "seller") {  
 NamedNodeMap attributes = flatChilds.item(i).getAttributes();  
 for (int k = 0; k < attributes.getLength(); k++) {  
 System.*out*.println(attributes.item(k).getNodeName());  
 if (attributes.item(k).getNodeName()=="title") {  
 sellerName = attributes.item(k).getNodeValue();  
 } else sellerPhone = attributes.item(k).getNodeValue();  
 }  
 break;  
 }  
 }  
 HashMap<String, Double> flatInfo = new HashMap();  
 NamedNodeMap attributes = flat.getAttributes();  
 for (int k = 0; k < attributes.getLength(); k++) {  
 flatInfo.put(attributes.item(k).getNodeName(), Double.*parseDouble*(attributes.item(k).getNodeValue()));  
 System.*out*.println(attributes.item(k).getNodeName()+" "+ Double.*parseDouble*(attributes.item(k).getNodeValue()));  
 }  
 ArrayList<Double> roomsSquare = new ArrayList<Double>();  
 for (int i = 0; i < flatChilds.getLength(); i++) {  
 if (flatChilds.item(i).getNodeName() == "rooms") {  
 NodeList rooms = flatChilds.item(i).getChildNodes();  
 for (int j = 0; j < rooms.getLength(); j++) {  
 roomsSquare.add(Double.*parseDouble*(rooms.item(i).getTextContent().trim()));  
 }  
 break;  
 }  
 }  
 Apartment apartment = new Apartment(sellerName, sellerPhone, region, flatInfo.get("square"), flatInfo.get("kitchen"), flatInfo.get("housingpart"), roomsSquare);  
 System.*out*.println(apartment.toString());  
 }  
}

Парсинг saxe

package com.company;  
  
import org.w3c.dom.Document;  
import org.w3c.dom.NamedNodeMap;  
import org.w3c.dom.Node;  
import org.w3c.dom.NodeList;  
import org.xml.sax.Attributes;  
import org.xml.sax.SAXException;  
import org.xml.sax.helpers.DefaultHandler;  
  
import javax.xml.parsers.\*;  
import java.awt.\*;  
import java.io.File;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.HashMap;  
  
public class Main {  
 public static class Apartment {  
 private String sellerName;  
 private String sellerPhoneNumber;  
 private String regionName;  
 private Double square;  
 private Double kitchenSquare;  
 private Double housingPartSquare;  
 private ArrayList<Double> roomsSquare = new ArrayList<Double>();  
  
 public Apartment(String sellerName, String sellerPhoneNumber, String regionName,  
 Double square, Double kitchenSquare, Double housingPartSquare,  
 ArrayList<Double> roomsSquare) {  
 this.sellerName = sellerName;  
 this.sellerPhoneNumber = sellerPhoneNumber;  
 this.regionName = regionName;  
 this.kitchenSquare = kitchenSquare;  
 this.square = square;  
 this.housingPartSquare = housingPartSquare;  
 this.roomsSquare = roomsSquare;  
 }  
 //<editor-fold desc="getters&setters">  
 public ArrayList<Double> getRoomsSquare() {  
 return roomsSquare;  
 }  
  
 public Double getHousingPartSquare() {  
 return housingPartSquare;  
 }  
  
 public Double getKitchenSquare() {  
 return kitchenSquare;  
 }  
  
 public Double getSquare() {  
 return square;  
 }  
  
 public String getRegionName() {  
 return regionName;  
 }  
  
 public String getSellerName() {  
 return sellerName;  
 }  
  
 public String getSellerPhoneNumber() {  
 return sellerPhoneNumber;  
 }  
  
 public void setHousingPartSquare(Double housingPartSquare) {  
 this.housingPartSquare = housingPartSquare;  
 }  
  
 public void setKitchenSquare(Double kitchenSquare) {  
 this.kitchenSquare = kitchenSquare;  
 }  
  
 public void setRegionName(String regionName) {  
 this.regionName = regionName;  
 }  
  
 public void setRoomsSquare(ArrayList<Double> roomsSquare) {  
 this.roomsSquare = roomsSquare;  
 }  
  
 public void setSellerName(String sellerName) {  
 this.sellerName = sellerName;  
 }  
  
 public void setSellerPhoneNumber(String sellerPhoneNumber) {  
 this.sellerPhoneNumber = sellerPhoneNumber;  
 }  
  
 public void setSquare(Double square) {  
 this.square = square;  
 }  
 //</editor-fold>  
 @Override  
 public final String toString() {  
 String \_roomsSquare = "";  
 for (var square : roomsSquare) {  
 \_roomsSquare += square + " ";  
 }  
 return " sellerName = " + sellerName + " sellerPhoneNumber = " + sellerPhoneNumber + " regionName = " + regionName +  
 " square = " + square.toString() + " kitchenSquare = " + kitchenSquare.toString() + " housingPartSquare = " + housingPartSquare.toString() +  
 " roomsSquare = " + \_roomsSquare;  
 }  
 }  
  
 private static class AdvancedXMLHandler extends DefaultHandler {  
 private String lastElementName;  
 private String sellerName;  
 private String sellerPhoneNumber;  
 private String lastRegionName;  
 private Double square;  
 private Double kitchenSquare;  
 private Double housingPartSquare;  
 private ArrayList<Double> roomsSquare = new ArrayList<Double>();  
  
 @Override  
 public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {  
 if (qName == "region") {  
 for (int i = 0; i < attributes.getLength(); i++) {  
 if (attributes.getQName(i) == "name" && attributes.getValue(i) != lastRegionName) {  
 lastRegionName = attributes.getValue(i);  
 }  
 }  
 }  
 if (qName == "seller") {  
 for (int i = 0; i < attributes.getLength(); i++) {  
 if (attributes.getQName(i) == "title") {  
 sellerName = attributes.getValue(i);  
 } else if (attributes.getQName(i) == "tel") {  
 sellerPhoneNumber = attributes.getValue(i);  
 }  
 }  
 }  
 if (qName == "flat") {  
 //count="2" square="51.2" kitchen="14.6" housingpart="30.6"  
 for (int i = 0; i < attributes.getLength(); i++) {  
 if (attributes.getQName(i) == "square") {  
 square = Double.*parseDouble*(attributes.getValue(i));  
 } else if (attributes.getQName(i) == "kitchen") {  
 kitchenSquare = Double.*parseDouble*(attributes.getValue(i));  
 } else if (attributes.getQName(i) == "housingpart") {  
 housingPartSquare = Double.*parseDouble*(attributes.getValue(i));  
 }  
 }  
 }  
 lastElementName = qName;  
 //System.out.println(qName);  
 }  
  
 @Override  
 public void characters(char[] ch, int start, int length) throws SAXException {  
 String information = new String(ch, start, length);  
  
 information = information.replace("\n", "").trim();  
 if (!information.isEmpty()) {  
 if (lastElementName == ("room"))  
 roomsSquare.add(Double.*parseDouble*(information.trim()));  
 }  
 }  
  
 @Override  
 public void endElement(String uri, String localName, String qName) throws SAXException {  
 if ((lastElementName != null && !lastElementName.isEmpty()) && (sellerName != null && !sellerName.isEmpty())  
 && (sellerPhoneNumber != null && !sellerPhoneNumber.isEmpty()) && (lastRegionName != null && !lastRegionName.isEmpty())  
 && (square != null) && (kitchenSquare != null)  
 && (housingPartSquare != null) && (roomsSquare != null && roomsSquare.size() > 0)) {  
 *apartments*.add(new Apartment(sellerName, sellerPhoneNumber, lastRegionName, square, kitchenSquare, housingPartSquare, roomsSquare));  
 sellerName = null;  
 sellerPhoneNumber = null;  
 square = null;  
 kitchenSquare = null;  
 housingPartSquare = null;  
 roomsSquare = new ArrayList<Double>();  
 }  
 }  
 }  
  
  
 private static ArrayList<Apartment> *apartments* = new ArrayList<>();  
  
 public static void main(String[] args) throws ParserConfigurationException, IOException, SAXException {  
 SAXParserFactory factory = SAXParserFactory.*newInstance*();  
 SAXParser parser = factory.newSAXParser();  
  
 AdvancedXMLHandler handler = new AdvancedXMLHandler();  
 parser.parse(new File("info.xml"), handler);  
  
 for (var apartment : *apartments*)  
 System.*out*.println(apartment.toString());  
 System.*out*.println(*apartments*.size());  
 }  
}

Парсинг sax + запись в xml

package com.company;  
  
import org.w3c.dom.\*;  
import org.xml.sax.Attributes;  
import org.xml.sax.SAXException;  
import org.xml.sax.helpers.DefaultHandler;  
  
import javax.xml.parsers.\*;  
import javax.xml.transform.Transformer;  
import javax.xml.transform.TransformerException;  
import javax.xml.transform.TransformerFactory;  
import javax.xml.transform.TransformerFactoryConfigurationError;  
import javax.xml.transform.dom.DOMSource;  
import javax.xml.transform.stream.StreamResult;  
import java.awt.\*;  
import java.io.File;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.HashSet;  
  
public class Main {  
 public static class Apartment {  
 private String sellerName;  
 private String sellerPhoneNumber;  
 private String regionName;  
 private Double square;  
 private Double kitchenSquare;  
 private Double housingPartSquare;  
 private ArrayList<Double> roomsSquare = new ArrayList<Double>();  
  
 public Apartment(String sellerName, String sellerPhoneNumber, String regionName,  
 Double square, Double kitchenSquare, Double housingPartSquare,  
 ArrayList<Double> roomsSquare) {  
 this.sellerName = sellerName;  
 this.sellerPhoneNumber = sellerPhoneNumber;  
 this.regionName = regionName;  
 this.kitchenSquare = kitchenSquare;  
 this.square = square;  
 this.housingPartSquare = housingPartSquare;  
 this.roomsSquare = roomsSquare;  
 }  
  
 //<editor-fold desc="getters&setters">  
 public ArrayList<Double> getRoomsSquare() {  
 return roomsSquare;  
 }  
  
 public Double getHousingPartSquare() {  
 return housingPartSquare;  
 }  
  
 public Double getKitchenSquare() {  
 return kitchenSquare;  
 }  
  
 public Double getSquare() {  
 return square;  
 }  
  
 public String getRegionName() {  
 return regionName;  
 }  
  
 public String getSellerName() {  
 return sellerName;  
 }  
  
 public String getSellerPhoneNumber() {  
 return sellerPhoneNumber;  
 }  
  
 public void setHousingPartSquare(Double housingPartSquare) {  
 this.housingPartSquare = housingPartSquare;  
 }  
  
 public void setKitchenSquare(Double kitchenSquare) {  
 this.kitchenSquare = kitchenSquare;  
 }  
  
 public void setRegionName(String regionName) {  
 this.regionName = regionName;  
 }  
  
 public void setRoomsSquare(ArrayList<Double> roomsSquare) {  
 this.roomsSquare = roomsSquare;  
 }  
  
 public void setSellerName(String sellerName) {  
 this.sellerName = sellerName;  
 }  
  
 public void setSellerPhoneNumber(String sellerPhoneNumber) {  
 this.sellerPhoneNumber = sellerPhoneNumber;  
 }  
  
 public void setSquare(Double square) {  
 this.square = square;  
 }  
  
 //</editor-fold>  
 @Override  
 public final String toString() {  
 String \_roomsSquare = "";  
 for (var square : roomsSquare) {  
 \_roomsSquare += square + " ";  
 }  
 return " sellerName = " + sellerName + " sellerPhoneNumber = " + sellerPhoneNumber + " regionName = " + regionName +  
 " square = " + square.toString() + " kitchenSquare = " + kitchenSquare.toString() + " housingPartSquare = " + housingPartSquare.toString() +  
 " roomsSquare = " + \_roomsSquare;  
 }  
 }  
  
 private static class AdvancedXMLHandler extends DefaultHandler {  
 private String lastElementName;  
 private String sellerName;  
 private String sellerPhoneNumber;  
 private String lastRegionName;  
 private Double square;  
 private Double kitchenSquare;  
 private Double housingPartSquare;  
 private ArrayList<Double> roomsSquare = new ArrayList<Double>();  
  
 @Override  
 public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {  
 if (qName == "region") {  
 for (int i = 0; i < attributes.getLength(); i++) {  
 if (attributes.getQName(i) == "name" && attributes.getValue(i) != lastRegionName) {  
 lastRegionName = attributes.getValue(i);  
 }  
 }  
 }  
 if (qName == "seller") {  
 for (int i = 0; i < attributes.getLength(); i++) {  
 if (attributes.getQName(i) == "title") {  
 sellerName = attributes.getValue(i);  
 } else if (attributes.getQName(i) == "tel") {  
 sellerPhoneNumber = attributes.getValue(i);  
 }  
 }  
 }  
 if (qName == "flat") {  
 for (int i = 0; i < attributes.getLength(); i++) {  
 if (attributes.getQName(i) == "square") {  
 square = Double.*parseDouble*(attributes.getValue(i));  
 } else if (attributes.getQName(i) == "kitchen") {  
 kitchenSquare = Double.*parseDouble*(attributes.getValue(i));  
 } else if (attributes.getQName(i) == "housingpart") {  
 housingPartSquare = Double.*parseDouble*(attributes.getValue(i));  
 }  
 }  
 }  
 lastElementName = qName;  
 }  
  
 @Override  
 public void characters(char[] ch, int start, int length) throws SAXException {  
 String information = new String(ch, start, length);  
  
 information = information.replace("\n", "").trim();  
 if (!information.isEmpty()) {  
 if (lastElementName == ("room"))  
 roomsSquare.add(Double.*parseDouble*(information.trim()));  
 }  
 }  
  
 @Override  
 public void endElement(String uri, String localName, String qName) throws SAXException {  
 if ((lastElementName != null && !lastElementName.isEmpty()) && (sellerName != null && !sellerName.isEmpty())  
 && (sellerPhoneNumber != null && !sellerPhoneNumber.isEmpty()) && (lastRegionName != null && !lastRegionName.isEmpty())  
 && (square != null) && (kitchenSquare != null)  
 && (housingPartSquare != null) && (roomsSquare != null && roomsSquare.size() > 0)) {  
 *apartments*.add(new Apartment(sellerName, sellerPhoneNumber, lastRegionName, square, kitchenSquare, housingPartSquare, roomsSquare));  
 sellerName = null;  
 sellerPhoneNumber = null;  
 square = null;  
 kitchenSquare = null;  
 housingPartSquare = null;  
 roomsSquare = new ArrayList<Double>();  
 }  
 }  
 }  
  
 private static void writeDocument(Document document, String fileName) throws TransformerFactoryConfigurationError {  
 try {  
 //объект класса Transformer может использоваться для записи результатов преобразования в различные приемники с помощью метода transform  
 Transformer tr = TransformerFactory  
 .*newInstance*()  
 .newTransformer();  
 //Объекты класса DOMSource используются для хранения исходного дерева преобразования в форме дерева объектной модели документа (DOM).  
 DOMSource source = new DOMSource(document);  
 FileOutputStream fos = new  
 FileOutputStream(fileName);  
 StreamResult result = new StreamResult(fos);  
 // метод transform(Source xmlSource, Result outputTarget) преобразует XML ресурс в какой-то из классов-наследников класса Result.  
 tr.transform(source, result);  
 } catch (TransformerException | IOException e) {  
 e.printStackTrace(System.*out*);  
 }  
 }  
  
  
 private static ArrayList<Apartment> *apartments* = new ArrayList<>();  
  
 public static void main(String[] args) throws ParserConfigurationException, IOException, SAXException {  
 SAXParserFactory factory = SAXParserFactory.*newInstance*();  
 SAXParser parser = factory.newSAXParser();  
 AdvancedXMLHandler handler = new AdvancedXMLHandler();  
 parser.parse(new File("info.xml"), handler);  
 HashSet<Integer> roomsCounts = new HashSet<Integer>();  
 for (var apartment : *apartments*) {  
 System.*out*.println(apartment.toString());  
 roomsCounts.add(apartment.getRoomsSquare().size());  
 }  
  
 DocumentBuilderFactory factory2 = DocumentBuilderFactory.*newInstance*();  
 factory2.setNamespaceAware(true);  
 Document doc = factory2.newDocumentBuilder().newDocument();  
 Element root = doc.createElement("supply");  
 doc.appendChild(root);  
 for (var roomsCount : roomsCounts) {  
 Element item1 = doc.createElement("item");  
 item1.setAttribute("roomcount", roomsCount.toString());  
 root.appendChild(item1);  
 for (var apartment : *apartments*) {  
 if (apartment.getRoomsSquare().size() == roomsCount) {  
 Element flat = doc.createElement("flat");  
 flat.setAttribute("square", apartment.getSquare().toString());  
 flat.setAttribute("kitchen", apartment.getKitchenSquare().toString());  
 flat.setAttribute("housingpart", apartment.getHousingPartSquare().toString());  
 flat.setAttribute("region", apartment.getRegionName());  
 item1.appendChild(flat);  
 Element rooms = doc.createElement("rooms");  
 flat.appendChild(rooms);  
 for (var square : apartment.getRoomsSquare()) {  
 Element \_square = doc.createElement("room");  
 \_square.setTextContent(square.toString());  
 rooms.appendChild(\_square);  
 }  
 Element seller = doc.createElement("seller");  
 seller.setAttribute("tel", apartment.getSellerPhoneNumber());  
 seller.setAttribute("title", apartment.getSellerName());  
 flat.appendChild(seller);  
 }  
 }  
 }  
 *writeDocument*(doc, "other.xml");  
 }  
}

package com.company;  
  
import org.w3c.dom.\*;  
import org.xml.sax.Attributes;  
import org.xml.sax.SAXException;  
import org.xml.sax.helpers.DefaultHandler;  
  
import javax.xml.parsers.\*;  
import javax.xml.transform.Transformer;  
import javax.xml.transform.TransformerException;  
import javax.xml.transform.TransformerFactory;  
import javax.xml.transform.TransformerFactoryConfigurationError;  
import javax.xml.transform.dom.DOMSource;  
import javax.xml.transform.stream.StreamResult;  
import java.awt.\*;  
import java.io.File;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.HashSet;  
import java.util.Scanner;  
  
public class Main {  
 public static class Apartment {  
 private String sellerName;  
 private String sellerPhoneNumber;  
 private String regionName;  
 private Double square;  
 private Double kitchenSquare;  
 private Double housingPartSquare;  
 private ArrayList<Double> roomsSquare = new ArrayList<Double>();  
  
 public Apartment(String sellerName, String sellerPhoneNumber, String regionName,  
 Double square, Double kitchenSquare, Double housingPartSquare,  
 ArrayList<Double> roomsSquare) {  
 this.sellerName = sellerName;  
 this.sellerPhoneNumber = sellerPhoneNumber;  
 this.regionName = regionName;  
 this.kitchenSquare = kitchenSquare;  
 this.square = square;  
 this.housingPartSquare = housingPartSquare;  
 this.roomsSquare = roomsSquare;  
 }  
  
 //<editor-fold desc="getters&setters">  
 public ArrayList<Double> getRoomsSquare() {  
 return roomsSquare;  
 }  
  
 public Double getHousingPartSquare() {  
 return housingPartSquare;  
 }  
  
 public Double getKitchenSquare() {  
 return kitchenSquare;  
 }  
  
 public Double getSquare() {  
 return square;  
 }  
  
 public String getRegionName() {  
 return regionName;  
 }  
  
 public String getSellerName() {  
 return sellerName;  
 }  
  
 public String getSellerPhoneNumber() {  
 return sellerPhoneNumber;  
 }  
  
 public void setHousingPartSquare(Double housingPartSquare) {  
 this.housingPartSquare = housingPartSquare;  
 }  
  
 public void setKitchenSquare(Double kitchenSquare) {  
 this.kitchenSquare = kitchenSquare;  
 }  
  
 public void setRegionName(String regionName) {  
 this.regionName = regionName;  
 }  
  
 public void setRoomsSquare(ArrayList<Double> roomsSquare) {  
 this.roomsSquare = roomsSquare;  
 }  
  
 public void setSellerName(String sellerName) {  
 this.sellerName = sellerName;  
 }  
  
 public void setSellerPhoneNumber(String sellerPhoneNumber) {  
 this.sellerPhoneNumber = sellerPhoneNumber;  
 }  
  
 public void setSquare(Double square) {  
 this.square = square;  
 }  
  
 //</editor-fold>  
  
 @Override  
 public final String toString() {  
 String \_roomsSquare = "";  
 for (var square : roomsSquare) {  
 \_roomsSquare += square + " ";  
 }  
 return " sellerName = " + sellerName + " sellerPhoneNumber = " + sellerPhoneNumber + " regionName = " + regionName +  
 " square = " + square.toString() + " kitchenSquare = " + kitchenSquare.toString() + " housingPartSquare = " + housingPartSquare.toString() +  
 " roomsSquare = " + \_roomsSquare;  
 }  
 }  
  
 private static class AdvancedXMLHandler extends DefaultHandler {  
 private String targetRegion;  
  
 public String getTargetRegion() {  
 return targetRegion;  
 }  
  
 public void setTargetRegion(String targetRegion) {  
 this.targetRegion = targetRegion;  
 }  
  
 private String lastElementName;  
 private String sellerName;  
 private String sellerPhoneNumber;  
 private String lastRegionName;  
 private Double square;  
 private Double kitchenSquare;  
 private Double housingPartSquare;  
 private ArrayList<Double> roomsSquare = new ArrayList<Double>();  
  
 @Override  
 public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {  
 if (qName == "region") {  
 for (int i = 0; i < attributes.getLength(); i++) {  
 if (attributes.getQName(i) == "name" && attributes.getValue(i) != lastRegionName) {  
 lastRegionName = attributes.getValue(i);  
 }  
 }  
 }  
 if (qName == "seller") {  
 for (int i = 0; i < attributes.getLength(); i++) {  
 if (attributes.getQName(i) == "title") {  
 sellerName = attributes.getValue(i);  
 } else if (attributes.getQName(i) == "tel") {  
 sellerPhoneNumber = attributes.getValue(i);  
 }  
 }  
 }  
 if (qName == "flat") {  
 for (int i = 0; i < attributes.getLength(); i++) {  
 if (attributes.getQName(i) == "square") {  
 square = Double.*parseDouble*(attributes.getValue(i));  
 } else if (attributes.getQName(i) == "kitchen") {  
 kitchenSquare = Double.*parseDouble*(attributes.getValue(i));  
 } else if (attributes.getQName(i) == "housingpart") {  
 housingPartSquare = Double.*parseDouble*(attributes.getValue(i));  
 }  
 }  
 }  
 lastElementName = qName;  
 }  
  
 @Override  
 public void characters(char[] ch, int start, int length) throws SAXException {  
 String information = new String(ch, start, length);  
  
 information = information.replace("\n", "").trim();  
 if (!information.isEmpty()) {  
 if (lastElementName == ("room"))  
 roomsSquare.add(Double.*parseDouble*(information.trim()));  
 }  
 }  
  
 @Override  
 public void endElement(String uri, String localName, String qName) throws SAXException {  
 if ((lastElementName != null && !lastElementName.isEmpty()) && (sellerName != null && !sellerName.isEmpty())  
 && (sellerPhoneNumber != null && !sellerPhoneNumber.isEmpty()) && (lastRegionName != null && !lastRegionName.isEmpty())  
 && (square != null) && (kitchenSquare != null)  
 && (housingPartSquare != null) && (roomsSquare != null && roomsSquare.size() > 0)) {  
 //apartments.add(new Apartment(sellerName, sellerPhoneNumber, lastRegionName, square, kitchenSquare, housingPartSquare, roomsSquare));  
 if (targetRegion == null)  
 *apartments*.add(new Apartment(sellerName, sellerPhoneNumber, lastRegionName, square, kitchenSquare, housingPartSquare, roomsSquare));  
 else if (targetRegion.equals(lastRegionName))  
 *apartments*.add(new Apartment(sellerName, sellerPhoneNumber, lastRegionName, square, kitchenSquare, housingPartSquare, roomsSquare));  
 sellerName = null;  
 sellerPhoneNumber = null;  
 square = null;  
 kitchenSquare = null;  
 housingPartSquare = null;  
 roomsSquare = new ArrayList<Double>();  
 }  
 }  
 }  
  
 private static void writeDocument(Document document, String fileName) throws TransformerFactoryConfigurationError {  
 try {  
 //объект класса Transformer может использоваться для записи результатов преобразования в различные приемники с помощью метода transform  
 Transformer tr = TransformerFactory  
 .*newInstance*()  
 .newTransformer();  
 //Объекты класса DOMSource используются для хранения исходного дерева преобразования в форме дерева объектной модели документа (DOM).  
 DOMSource source = new DOMSource(document);  
 FileOutputStream fos = new  
 FileOutputStream(fileName);  
 StreamResult result = new StreamResult(fos);  
 // метод transform(Source xmlSource, Result outputTarget) преобразует XML ресурс в какой-то из классов-наследников класса Result.  
 tr.transform(source, result);  
 } catch (TransformerException | IOException e) {  
 e.printStackTrace(System.*out*);  
 }  
 }  
  
  
 private static ArrayList<Apartment> *apartments* = new ArrayList<>();  
  
 public static void main(String[] args) throws ParserConfigurationException, IOException, SAXException {  
 SAXParserFactory factory = SAXParserFactory.*newInstance*();  
 SAXParser parser = factory.newSAXParser();  
 AdvancedXMLHandler handler = new AdvancedXMLHandler();  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введите регион: ");  
 String targetRegion= scanner.nextLine();  
 handler.setTargetRegion(targetRegion);  
 parser.parse(new File("info.xml"), handler);  
 HashSet<Integer> roomsCounts = new HashSet<Integer>();  
 for (var apartment : *apartments*) {  
 System.*out*.println(apartment.toString());  
 roomsCounts.add(apartment.getRoomsSquare().size());  
 }  
 }  
}