Jegyzőkönyv

Adatkezelés XML környezetben

Féléves feladat

Taxi rendelési rendszer

Készítette: Bartók Peter László

Neptunkód: RVQUE9

Dátum:2024. december 10.

Tartalom

[A feladat leírása 3](#_Toc184828283)

[Szereplők és funkciók 3](#_Toc184828284)

[a. Utasok 3](#_Toc184828285)

[b. Rendelések 3](#_Toc184828286)

[c. Út 3](#_Toc184828287)

[d. Vezetők 4](#_Toc184828288)

[e. Taxik 4](#_Toc184828289)

[f. Kapcsolatok a rendszerben 4](#_Toc184828290)

[g. Rendszer előnyei 5](#_Toc184828291)

[Az adatbázis ER modell tervezése 5](#_Toc184828292)

[h. Követelmények elemzése 5](#_Toc184828293)

[i. Entitások és attribútumaik azonosítása 5](#_Toc184828294)

[j. Kapcsolatok azonosítása 6](#_Toc184828295)

[k. Kulcsok meghatározása 6](#_Toc184828296)

[l. Kapcsolati típusok pontosítása 7](#_Toc184828297)

[Az adatbázis konvertálása XDM modellre 7](#_Toc184828298)

[Az XDM modell alapján XML dokumentum készítése 8](#_Toc184828299)

[m. XML kód: 8](#_Toc184828300)

1. Bevezetés

A választott feladat egy taxirendelési rendszer adatainak kezelése. Segíti az utasok, rendelések, utak, vezetők és taxik közötti kapcsolatok rendszerezését és hatékony nyomon követését.

## A feladat leírása

A Taxi rendelési rendszer célja, hogy hatékonyan kezelje a taxirendelésekhez kapcsolódó adatokat, és biztosítsa a különböző szereplők – például utasok, vezetők és taxi járművek – közötti információáramlást. Az alábbiakban részletezem a rendszer működését és logikai felépítését.

1. Rendszer célja

A rendszer a taxirendelési folyamat minden részét lefedi:

**Utasok kezelése**: Az utasok adatainak nyilvántartása, rendeléseik és preferenciáik nyomon követése.

**Rendelések menedzselése**: Minden taxirendelés adatainak rögzítése, beleértve a felvételi és leadási helyet, a rendelési időpontot és a díjazást.

**Útkezelés**: Az egyes rendelésekhez tartozó útvonalak, távolságok és utazási idő követése.

**Vezetők és taxik nyilvántartása**: A sofőrök jogosítványainak és a taxik részletes adatainak kezelése.

## Szereplők és funkciók

## Utasok

Az utasok a rendszer központi szereplői, akik rendeléseket hoznak létre. A rendszer rögzíti:

Az utasok személyes adatait, beleértve nevüket, elérhetőségeiket (telefon, e-mail) és lakcímüket.

A lakcím összetett adatként tárolódik, amely magában foglalja a várost, utcát és házszámot, valamint az irányítószámot. Példa használat: Egy utas regisztrálhat a rendszerbe, majd több rendelést is leadhat különböző időpontokban.

## Rendelések

Minden rendelés tartalmazza azokat az adatokat, amelyek szükségesek egy taxifuvar megszervezéséhez:

A felvételi helyet és a leadási helyet, ahol az utast felveszik és ahova szállítják. A rendelési időpontot, amely segít a későbbi elemzésekben és szervezésben. A fuvar tarifáját, amely a távolság és az idő alapján kerül meghatározásra. Példa használat: Egy rendelés során az utas kiválasztja, honnan szeretné elindítani az utazást (pl. otthonából), és megadja a célállomást.

## Út

Az egyes rendelésekhez tartozó útvonalak részletesen nyilván vannak tartva:

* Az út kiindulópontja és végpontja, amelyek meghatározzák az utazás helyszíneit.
* Az út távolsága és a várható utazási idő, amely elősegíti a pontos díjszámítást.

Példa használat: A rendszer kiszámítja, hogy egy adott rendeléshez tartozó út 15 kilométer hosszú, és körülbelül 20 percet vesz igénybe.

## Vezetők

A rendszerben szereplő vezetők azok, akik a rendeléseket teljesítik. A vezetőkről tárolt adatok a következők:

* Személyes adatok: név, telefonszám.
* Jogosítvánnyal kapcsolatos információk: a jogosítvány száma és kategóriája (pl. B kategória).
* Az általuk üzemeltetett jármű, amely kapcsolódik egy konkrét taxijárműhöz.

Példa használat: Egy vezető elérhetőségei és jogosítványának érvényessége alapján ellenőrizhető a fuvarra való alkalmassága.

## Taxik

A taxik fizikai járművek, amelyekhez vezetők és rendelések kapcsolódnak. Az adatbázis tárolja:

* A taxi rendszámát, amely az egyedi azonosítás alapja.
* A jármű modelljét, amely segít a flotta kezelésében.
* A kapacitást (hány utas szállítható), amely alapján a rendeléseket optimalizálják.

Példa használat: Egy taxi maximális kapacitása 4 utas. Ha egy utas nagyobb csoportot szeretne szállíttatani, a rendszer figyelmeztetést adhat.

## Kapcsolatok a rendszerben

*Utas és Rendelés kapcsolata:*

Egy utas több rendelést is leadhat.

A rendszerben az "Utas ID" segítségével kapcsolódnak az utasok a rendeléseikhez.

*Rendelés és Út kapcsolata:*

Minden rendeléshez tartozik egy útvonal.

Az "Út ID" azonosítja, hogy melyik útra vonatkozik az adott rendelés.

*Út és Vezető kapcsolata:*

Egy vezető teljesít egy adott útvonalat.

A kapcsolat a "Vezető ID" segítségével jön létre.

*Vezető és Taxi kapcsolata:*

Egy vezető egy taxit üzemeltet az adott fuvar során.

A kapcsolatot a "Taxi ID" biztosítja.

## Rendszer előnyei

**Hatékonyság:** Az utasok, vezetők és taxik közötti kapcsolatok automatizált kezelése csökkenti az adminisztratív terheket.

**Nyomon követhetőség:** A rendelések, utak és vezetők adatai egyértelműen kapcsolódnak egymáshoz.

**Testreszabhatóság:** A tárolt adatok egyszerűen bővíthetők, például extra funkciókkal (pl. utas preferenciák).

**Pontos díjszámítás:** Az utak távolsága és időtartama alapján pontos tarifa számítható.

1. 1.Feadat

## Az adatbázis ER modell tervezése

Az ER modell (Entitás-Kapcsolat modell) tervezése egy adatbázis-rendszer kialakításának egyik kulcsfontosságú lépése. Az alábbiakban bemutatom a tervezési lépéseket egy példával, amely egy taxirendelési rendszerre vonatkozik.

## Követelmények elemzése

Ez a lépés a rendszer céljainak és az üzleti igényeknek a meghatározásával kezdődik. Meg kell érteni:

Kik a rendszer fő szereplői (utasok, vezetők, rendeléskezelők).

Milyen adatokra van szükség (utas adatai, rendelési információk, utak részletei stb.).

Hogyan kapcsolódnak az adatok egymáshoz (pl. egy utas több rendelést is leadhat).

Példa: Az ügyfél egy taxirendelési rendszert szeretne, amely kezeli az utasokat, rendeléseket, útvonalakat, vezetőket és taxikat.

## Entitások és attribútumaik azonosítása

Az entitások azok az objektumok, amelyek adatait tárolni kell. Az attribútumok az entitások jellemzői.

Azonosítás lépései:

* Határozzuk meg a fő objektumokat.
* Soroljuk fel az objektumok jellemzőit.

Példa:

Entitások:

* Utas (utasok adatai).
* Rendelés (minden rendeléshez tartozó részletek).
* Út (az utazás adatai).
* Vezető (a sofőrök adatai).
* Taxi (a járművek adatai).

Attribútumok példák:

Utas: Utas ID, név, telefon, e-mail, lakcím (összetett attribútum: város, utca, házszám, irányítószám).

Rendelés: Rendelés ID, felvétel helye, leadás helye, rendelési időpont, tarifa.

## Kapcsolatok azonosítása

Az entitások közötti kapcsolatok feltérképezése. Ezek lehetnek:

Egy az egyhez (1:1).

Egy a többhöz (1:N).

Több a többhöz (M:N).

Kapcsolatok megtervezése:

Határozzuk meg, hogyan kapcsolódnak az entitások.

Milyen kulcsokat (elsődleges és idegen kulcsokat) használunk a kapcsolatok létrehozásához.

Példa:

Egy utas több rendelést adhat le (Utas ↔ Rendelés, 1:N kapcsolat).

Egy rendelés egy adott útvonalhoz tartozik (Rendelés ↔ Út, 1:1 kapcsolat).

Egy vezető egy adott út során egy taxit vezet (Vezető ↔ Taxi, 1:1 kapcsolat).

## Kulcsok meghatározása

Minden entitásnak szüksége van egy elsődleges kulcsra (primary key), amely egyedi módon azonosítja az entitás rekordjait. Továbbá, az idegen kulcsok (foreign keys) biztosítják az entitások közötti kapcsolatok létrehozását.

Példa:

Elsődleges kulcsok:

* Utas: Utas ID.
* Rendelés: Rendelés ID.
* Út: Út ID.
* Vezető: Vezető ID.
* Taxi: Taxi ID.

Idegen kulcsok:

* Rendelés kapcsolódik az Utashoz az Utas ID-n keresztül.
* Út kapcsolódik a Rendeléshez az Út ID-n keresztül.
* Vezető kapcsolódik a Taxihoz az Taxi ID-n keresztül.

## Kapcsolati típusok pontosítása

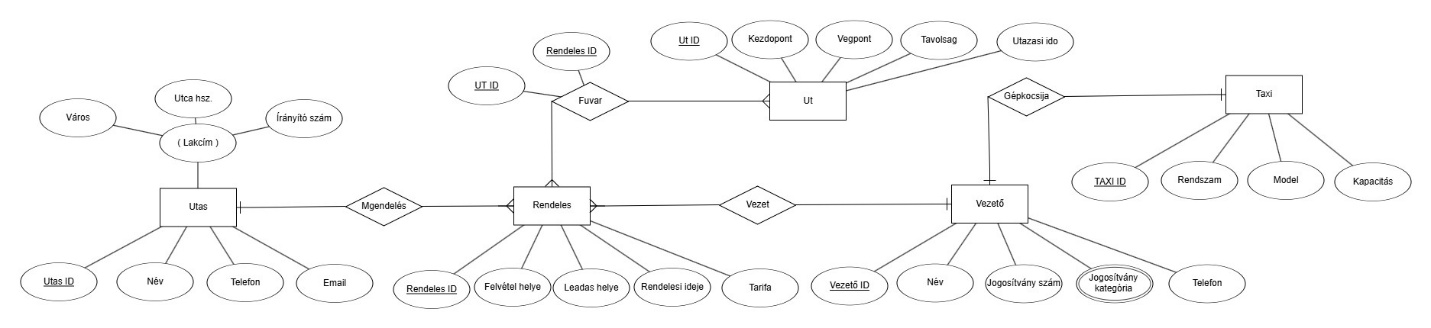
A kapcsolatok részletes kidolgozása, beleértve az egyedhalmazok közötti pontos kapcsolatokat (pl. teljes vagy részleges részvétel, kapcsolati kardinalitások).

Kardinalitások:

Egy utasnak több rendelése lehet, de minden rendelés egy konkrét utashoz tartozik (1:N).

Egy rendelés egyetlen útvonalhoz kapcsolódik (1:1).

Egy vezető egy adott időben egy taxit vezet (1:1).



Kép1-ER modell

## Az adatbázis konvertálása XDM modellre

Az XDM (XML Data Model) egy hierarchikus adatmodell, amely az XML dokumentumok fa alapú szerkezetét írja le. Ez az adatmodell az XML dokumentumok logikai felépítésének és tartalmának reprezentációjára szolgál. Az XDM nemcsak az XML fájlok szerkezetét, hanem az olyan kapcsolódó technológiák alapját is képezi, mint az XPath, XQuery és XSLT.

XDM modell magyarázata

Root Node (Taxi rendelés): A fő elem, amely tartalmazza az összes adatot a rendelésről.

Rendelés ID: Az egyedi rendelésazonosító, amely egy numerikus értéket tartalmaz.

Utas ID: A vevő adatait tartalmazza, beleértve az ügyfél azonosítóját, nevét, telefonszámát és email címét.

Út ID: A felvételi hely információit tartalmazza, mint cím, időpont és koordináták.

Rendelés ID: Az úti cél adatokat tartalmazza, mint cím és koordináták.

Gépkocsi ID: A jármű információit tartalmazza, mint a jármű azonosítója, típusa és rendszáma.

A képen vázlat, rajz, diagram, fehér látható

Automatikusan generált leírás

2.Ábra-XDM modell

## Az XDM modell alapján XML dokumentum készítése

A Taxirendelés XDM modell egy olyan hierarchikus adatmodellt jelent, amely leírja a taxirendelési folyamatot egy XML dokumentumban. Az XDM (XML Data Model) fa alapú struktúrája segít a taxirendelési információk pontos reprezentálásában, beleértve a rendelést, az ügyfelet, a járművet és a különböző egyéb adatokat.

A taxirendelés XDM modell egy tipikus példája annak, hogyan lehet a rendelési adatokat hierarchikus formában strukturálni, ahol minden egyes adat elem vagy csomópont a rendelés egy részét reprezentálja.

## XML kód:

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

<TaxiRendszer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="XMLSchemaRVQUE9.xsd">

<!-- Utas adatai-->

<Utas>

<UtasID>01</UtasID>

<Nev>Kiss István</Nev>

<Telefon>06702593123</Telefon>

<Email>kis.istvan@valami.com</Email>

<Lakcim>

<Varos>Gyula</Varos>

<UtcaHsz>Kossuth Lajos u. 18</UtcaHsz>

<IranyitoSzam>5700</IranyitoSzam>

</Lakcim>

</Utas>

<Utas>

<UtasID>02</UtasID>

<Nev>Németh Zoltán</Nev>

<Telefon>06305874236</Telefon>

<Email>zoltan.nemeth@upc.com</Email>

<Lakcim>

<Varos>Debrecen</Varos>

<UtcaHsz>Gázvezeték utca 85</UtcaHsz>

<IranyitoSzam>4030</IranyitoSzam>

</Lakcim>

</Utas>

<Utas>

<UtasID>03</UtasID>

<Nev>Balogh Antal</Nev>

<Telefon>06207846218</Telefon>

<Email>antal.balogh@t-online.hu</Email>

<Lakcim>

<Varos>Miskolc</Varos>

<UtcaHsz>Vörösmarty utca 52.</UtcaHsz>

<IranyitoSzam>3532</IranyitoSzam>

</Lakcim>

</Utas>

<Rendeles>

<RendelesID>001</RendelesID>

<FelvetelHelye>Miskolc</FelvetelHelye>

<LeadasHelye>Miskolc</LeadasHelye>

<RendelesIdeje>2024-11-12T13:20:00</RendelesIdeje>

<Tarifa>244</Tarifa>

<UtasID>03</UtasID>

<UtID>01</UtID>

</Rendeles>

<Rendeles>

<RendelesID>002</RendelesID>

<FelvetelHelye>Debrecen</FelvetelHelye>

<LeadasHelye>Debrecen</LeadasHelye>

<RendelesIdeje>2024-12-01T18:20:00</RendelesIdeje>

<Tarifa>244</Tarifa>

<UtasID>02</UtasID>

<UtID>02</UtID>

</Rendeles>

<Rendeles>

<RendelesID>003</RendelesID>

<FelvetelHelye>Gyula</FelvetelHelye>

<LeadasHelye>Gyula</LeadasHelye>

<RendelesIdeje>2004-12-06T17:53:00</RendelesIdeje>

<Tarifa>244</Tarifa>

<UtasID>01</UtasID>

<UtID>03</UtID>

</Rendeles>

<Ut>

<UtID>01</UtID>

<Kezdopont>Vörömarty utca 52.</Kezdopont>

<Vegpont>Győrikapu</Vegpont>

<Tavolsag>12</Tavolsag>

<UtasIdo>13</UtasIdo>

</Ut>

<Ut>

<UtID>02</UtID>

<Kezdopont>Piac utca</Kezdopont>

<Vegpont>Gázvezeték utca 85.</Vegpont>

<Tavolsag>9</Tavolsag>

<UtasIdo>12</UtasIdo>

</Ut>

<Ut>

<UtID>03</UtID>

<Kezdopont>Diófa utca 12.</Kezdopont>

<Vegpont>Kossuth Lajos u. 18</Vegpont>

<Tavolsag>9</Tavolsag>

<UtasIdo>16</UtasIdo>

</Ut>

<Vezeto>

<VezetoID>01</VezetoID>

<Nev>Hoffman Géza</Nev>

<Telefon>06208563179</Telefon>

<JogositvanySzam>CH135874</JogositvanySzam>

<JogositvanyKategoria>B</JogositvanyKategoria>

<Gepkocsija>

<TaxiID>01</TaxiID>

</Gepkocsija>

</Vezeto>

<Vezeto>

<VezetoID>02</VezetoID>

<Nev>Kelemen János</Nev>

<Telefon>06709517532</Telefon>

<JogositvanySzam>CT157893</JogositvanySzam>

<JogositvanyKategoria>B</JogositvanyKategoria>

<Gepkocsija>

<TaxiID>02</TaxiID>

</Gepkocsija>

</Vezeto>

<Vezeto>

<VezetoID>03</VezetoID>

<Nev>Horváth Péter</Nev>

<Telefon>06302874695</Telefon>

<JogositvanySzam>CD187496</JogositvanySzam>

<JogositvanyKategoria>B</JogositvanyKategoria>

<Gepkocsija>

<TaxiID>03</TaxiID>

</Gepkocsija>

</Vezeto>

<Taxi>

<TaxiID>01</TaxiID>

<Rendszam>AK-HT-578</Rendszam>

<Model>SKODA OCTAVIA Combi</Model>

<Kapacitas>5</Kapacitas>

</Taxi>

<Taxi>

<TaxiID>02</TaxiID>

<Rendszam>HJD-258</Rendszam>

<Model>VW GOLF Combi</Model>

<Kapacitas>5</Kapacitas>

</Taxi>

<Taxi>

<TaxiID>03</TaxiID>

<Rendszam>AA-FN-852</Rendszam>

<Model>AUDI A4</Model>

<Kapacitas>5</Kapacitas>

</Taxi>

</TaxiRendszer>

1. 2. feladat

Project name: DOMParseNeptunkod

Package: hu.domparse.neptunkod

Class names: (DomReadNeptunkod, DomModifyNeptunkod, DomQueryNeptunkod, DOMWriteNeptunkod)

1. adatolvasás–fájlnév: DOMReadRVQUE9.java

package hu.domparse.rvque9;

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

import java.io.File;

public class DOMReadRVQUE9 {

public static void main(String[] args) {

try {

// Create a DocumentBuilderFactory

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

// Create a DocumentBuilder

DocumentBuilder builder = factory.newDocumentBuilder();

// Parse the XML file and build the Document

Document document = builder.parse(new File("XMLRVQUE9.xml"));

// Normalize the XML structure

document.getDocumentElement().normalize();

// Print the root element

System.out.println("Root element: " + document.getDocumentElement().getNodeName());

// Get all Taxi elements

NodeList nodeList = document.getElementsByTagName("Taxi");

// Iterate through the nodes and print the details

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

Node node = nodeList.item(i);

if (node.getNodeType() == Node.ELEMENT\_NODE) {

Element element = (Element) node;

System.out.println("TaxiID: " + element.getElementsByTagName("TaxiID").item(0).getTextContent());

System.out.println("Rendszam: " + element.getElementsByTagName("Rendszam").item(0).getTextContent());

System.out.println("Model: " + element.getElementsByTagName("Model").item(0).getTextContent());

System.out.println("Kapacitas: " + element.getElementsByTagName("Kapacitas").item(0).getTextContent());

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

1. adatmódosítás–fájlnév: DOMModifyRQUE9.java

package hu.domparse.rvque9;

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

import javax.xml.transform.\*;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.io.File;

public class DOMModifyRVQUE9 {

public static void main(String[] args) {

try {

// Create a DocumentBuilderFactory

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

// Create a DocumentBuilder

DocumentBuilder builder = factory.newDocumentBuilder();

// Parse the XML file and build the Document

Document document = builder.parse(new File("XMLRVQUE9.xml"));

// Normalize the XML structure

document.getDocumentElement().normalize();

// Modify specific data

NodeList nodeList = document.getElementsByTagName("Taxi");

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

Node node = nodeList.item(i);

if (node.getNodeType() == Node.ELEMENT\_NODE) {

Element element = (Element) node;

String taxiID = element.getElementsByTagName("TaxiID").item(0).getTextContent();

if (taxiID.equals("03")) {

element.getElementsByTagName("Model").item(0).setTextContent("AUDI A6");

}

}

}

// Write the modified document to a new file

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(document);

StreamResult result = new StreamResult(new File("XMLRVQUE9\_modified.xml"));

transformer.transform(source, result);

System.out.println("XML file updated successfully");

} catch (Exception e) {

e.printStackTrace();

}

}

}

1. adatlekérdezés–fájlnév: RVQUE9.java

package hu.domparse.rvque9;

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

import java.io.File;

public class DOMQueryRVQUE9 {

public static void main(String[] args) {

try {

// Create a DocumentBuilderFactory

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

// Create a DocumentBuilder

DocumentBuilder builder = factory.newDocumentBuilder();

// Parse the XML file and build the Document

Document document = builder.parse(new File("XMLRVQUE9.xml"));

// Normalize the XML structure

document.getDocumentElement().normalize();

// Query specific data

NodeList nodeList = document.getElementsByTagName("Taxi");

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

Node node = nodeList.item(i);

if (node.getNodeType() == Node.ELEMENT\_NODE) {

Element element = (Element) node;

String taxiID = element.getElementsByTagName("TaxiID").item(0).getTextContent();

if (taxiID.equals("02")) {

System.out.println("TaxiID: " + taxiID);

System.out.println("Rendszam: " + element.getElementsByTagName("Rendszam").item(0).getTextContent());

System.out.println("Model: " + element.getElementsByTagName("Model").item(0).getTextContent());

System.out.println("Kapacitas: " + element.getElementsByTagName("Kapacitas").item(0).getTextContent());

}

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

1. adatírás - készítsen egy DOM API programot, amely egy XMLNeptunkod.xml dokumentum tartalmát fa struktúra formában kiírja a konzolra és egy XMLNeptunkod1.xml fájlba. fájlnév: DOMWriteRVQUE9.java

package hu.domparse.rvque9;

import org.w3c.dom.\*;

import javax.xml.parsers.\*;

import javax.xml.transform.\*;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.io.File;

public class DOMWriteRVQUE9 {

public static void main(String[] args) {

try {

// Create a DocumentBuilderFactory

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

// Create a DocumentBuilder

DocumentBuilder builder = factory.newDocumentBuilder();

// Parse the XML file and build the Document

Document document = builder.parse(new File("XMLRVQUE9.xml"));

// Normalize the XML structure

document.getDocumentElement().normalize();

// Write the document content to the console

printDocument(document, System.out);

// Write the document content to a new file

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(document);

StreamResult result = new StreamResult(new File("XMLRVQUE91.xml"));

transformer.transform(source, result);

System.out.println("XML file written successfully");

} catch (Exception e) {

e.printStackTrace();

}

}

private static void printDocument(Document doc, java.io.OutputStream out) throws Exception {

TransformerFactory tf = TransformerFactory.newInstance();

Transformer transformer = tf.newTransformer();

transformer.setOutputProperty(OutputKeys.OMIT\_XML\_DECLARATION, "no");

transformer.setOutputProperty(OutputKeys.METHOD, "xml");

transformer.setOutputProperty(OutputKeys.INDENT, "yes");

transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");

transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");

transformer.transform(new DOMSource(doc), new StreamResult(new java.io.OutputStreamWriter(out, "UTF-8")));

}

}