JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Webshop

Készítette: **Kerekes Krisztofer**

Neptunkód: **TRNA8A**

Dátum: **2022.11.29**

Tartalomjegyzék

A feladat leírása

Ebben a feladatban egy webshop adatbázisával dolgozom, ami tech termékeket árul. A webshop csak futárcéggel való kiszállítást biztosít, nincs lehetőség személyes átvételre. (telefon, laptop, tv, okosóra, stb.) Az XML fájlban tárolom el a vásárlók adatait, ez magában foglalja a vásárló teljes nevét (vezetéknév, keresztnév), a vásárló címét (város, utca, házszám), a vásárló telefonszámát és egy egyedi azonosítót, ami attribútumként jelenik meg. A fájlban szerepelnek még a termékek adatai, vagyis a termék elnevezése, rövid leírása, ára és a termék azonosítója, szintén attribútumként. Ezen felül a webshop partneri kapcsolatban áll futárcégekkel. Ezeknek a futárcégeknek az adatait is tartalmazza az XML fájl. Egy futárcégnek tárolom a nevét, címét, átlagos szállítási idejét, szállítási árát és egy általam hozzá rendelt azonosítót. A feladatban helyet kapott még egy raktárnyilvántartás is, ami a termékek raktáron lévő mennyiségét adja meg. Ez 1:1 kapcsolatban áll a termékekkel. Végül a rendelések is tárolva vannak. Egy rendelésnek van dátuma, ami év, hónap, nap, óra, perc mezőkből áll. Ezen felül egy rendeléshez tartoznak árucikkek (1 vagy több), aminek két mezője van, az egyik egy idegen kulcs, ami az adott terméket azonosítja, a másik a rendelt mennyiséget határozza meg. Minden rendeléshez tartozik egy futárcég, amit a megrendelő választ ki. A rendelésen belül tárolom a futárcég idegen kulcsát, ami azonosítja a futárcéget, illetve el van tárolva a felvétel ideje és a várható kézbesítés ideje. A rendeléseknek van egy egyedi azonosítójuk, ami attribútumként van definiálva és szintén attribútum ként szerepel egy idegen kulcs, ami a vásárlót azonosítja.

1. feladat

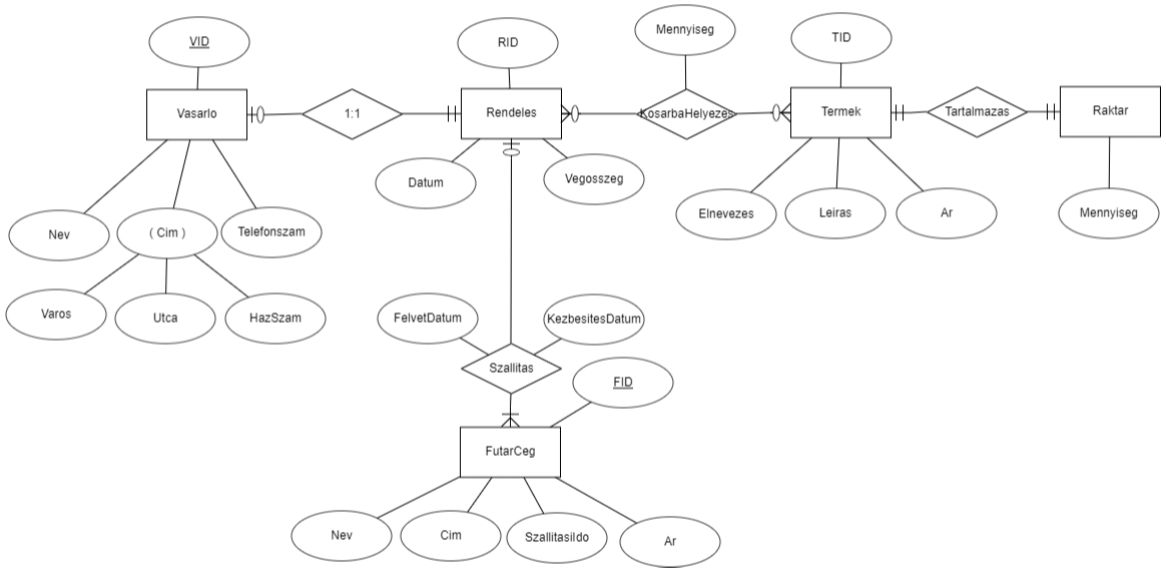
a) Az adatbázis ER modell

Az ER modellben 5 egyed szerepel, ezek a következők: *'Varsarlo'*, *'Rendeles'*, *'Termek'*, *'Raktar'*, *'Futarceg'*

Mindegyik egyednek saját egyedi azonosítója, kivéve a 'Raktar' egyednek.

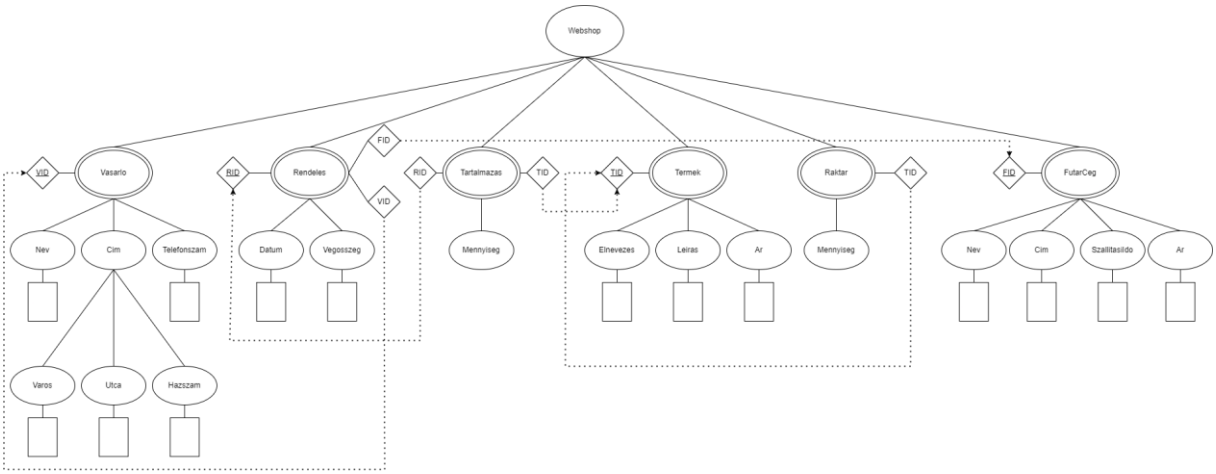
A *'Vasarlo'* egyednek 4 tulajdonsága van, ebből az egyik (cím) egy összetett tulajdonság. (Varos, Utca, Hazszam) A *'Termek'* egyednek 4, a *'Futarceg'* egyednek 5, a *'Raktar'* egyednek 1 és a *'Rendeles'* egyednek 3 tulajdonsága van.

- *'Vasarlo'* és a *'Rendeles'* között 1:N kapcsolat
- *'Rendeles'* és a *'Termek'* között N:M kapcsolat
- *'Termek'* és a *'Raktar'* között 1:1 kapcsolat
- *'Rendeles'* és a *'Futarceg'* között N:M kapcsolat



b) Az adatbázis konvertálása XDM modellre

Az XDM modellre való konvertáláskor minden egyedhez hozzárendeltem az idegen kulcsokat és mivel egy rendeléshez több termék is tartozhat, ezért bevezettem a *'Tartalmazas'* egyedet, mely összekapcsolja ezt a két egyedet és meghatározza a mennyiséget is.



c) Az XDM modell alapján XML dokumentum készítése

Az XML fájl felépítését a feladat leírásában részleteztem.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<webshop xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" xs:noNamespaceSchemaLocation="XMLSchemaTRNA8A.xsd">

    <!-- Vásárlók -->
    <vasarlo vid="v001">
        <vezeteknev>Kerekes</vezeteknev>
        <keresztnev>Krisztofer</keresztnev>
        <cim>
            <varos>Nyékládháza</varos>
            <utca>Balassi Bálint</utca>
            <hazszam>37</hazszam>
        </cim>
        <telefonszam>06706102436</telefonszam>
    </vasarlo>

    <!-- Termékek -->
    <termek tid="t001">
        <elnevezes>Samsung UE43TU7022 TV</elnevezes>
        <leiras>Merülj el a képben a szélesebb színskálával! A Crystal Display biztosítja az optimalizált színkifejezés
        <ar>129899</ar>
    </termek>
    <termek tid="t002">
        <elnevezes>Apple iPhone 13 128GB Mobiltelefon</elnevezes>
        <leiras>Az ultraszéles látószögű kamera közvetlen közelről is nagy látóteret ad, hogy hátralépés nélkül is több
        <ar>312970</ar>
    </termek>
    <termek tid="t003">
        <elnevezes>Samsung Galaxy A52s 5G 128GB</elnevezes>
        <leiras>A következő generációs mobiladat-hálózaton az 5G sebesség megváltoztatja a tapasztalatok és a tartalom
        <ar>136990</ar>
    </termek>
    <termek tid="t004">

<!-- Futárcégek -->
<futarceg fid="f01">
    <nev>FOXPOST</nev>
    <cim>1097 Budapest, Könyves Kálmán krt. 12-14</cim>
    <szallitasiido>2 nap</szallitasiido>
    <ar>1499</ar>
</futarceg>
<futarceg fid="f02">
    <nev>GLS</nev>
    <cim>Budapest</cim>
    <szallitasiido>3 nap</szallitasiido>
    <ar>1799</ar>
</futarceg>
<futarceg fid="f03">
    <nev>Packeta</nev>
    <cim>Budapest Ezred utca 2</cim>
    <szallitasiido>4 nap</szallitasiido>
    <ar>2000</ar>
</futarceg>


<!-- Raktár -->
<nyilvantartas tid_f="t001">
    <mennyiseg>15</mennyiseg>
</nyilvantartas>
<nyilvantartas tid_f="t002">
    <mennyiseg>6</mennyiseg>
</nyilvantartas>
<nyilvantartas tid_f="t003">
    <mennyiseg>32</mennyiseg>
</nyilvantartas>
<nyilvantartas tid_f="t004">
    <mennyiseg>24</mennyiseg>
</nyilvantartas>
<nyilvantartas tid_f="t005">
    <mennyiseg>13</mennyiseg>
</nyilvantartas>
```

```xml
    <!-- Rendelések -->
    <rendeles rid="r001" vid_f="v001">
        <datum>
            <ev>2022</ev>
            <honap>11</honap>
            <nap>03</nap>
            <ora>14</ora>
            <perc>32</perc>
        </datum>
        <tartalom>
            <arucikk>
                <tid_f>t002</tid_f>
                <mennyiseg>1</mennyiseg>
            </arucikk>
            <arucikk>
                <tid_f>t003</tid_f>
                <mennyiseg>2</mennyiseg>
            </arucikk>
        </tartalom>
        <vegosszeg>314469</vegosszeg>
        <futar>
            <fid_f>f01</fid_f>
            <felvetel>2022-11-04</felvetel>
            <kezbesites>2022-11-05</kezbesites>
        </futar>
    </rendeles>
</webshop>
```

d) Az XML dokumentum alapján XMLSchema készítése

Az XML fájl validálására létrehoztam az XMLSchema-t. Az IntelliJ fejlesztőkörnyezet automatikus validálást végzett.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="webshop">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="vasarlo" type="vasarloTipus" maxOccurs="unbounded"/>
                <xs:element name="termek" type="termekTipus" maxOccurs="unbounded"/>
                <xs:element name="futarceg" type="futarcegTipus" maxOccurs="unbounded"/>
                <xs:element name="nyilvantartas" type="nyilvantartasTipus" maxOccurs="unbounded"/>
                <xs:element name="rendeles" type="rendelesTipus" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>

        <!-- Elsődleges kulcsok -->
        <xs:key name="vasarloKulcs">
            <xs:selector xpath="vasarlo"/>
            <xs:field xpath="@vid"/>
        </xs:key>

        <xs:key name="termekKulcs">
            <xs:selector xpath="termek"/>
            <xs:field xpath="@tid"/>
        </xs:key>

        <xs:key name="futarcegKulcs">
            <xs:selector xpath="futarceg"/>
            <xs:field xpath="@fid"/>
        </xs:key>

        <xs:key name="rendelesKulcs">
            <xs:selector xpath="rendeles"/>
            <xs:field xpath="@rid"/>
        </xs:key>
```

```
<!-- Idegen kulcsok -->
<xs:keyref refer="vasarloKulcs" name="rendelesVasarloIdegenKulcs">
    <xs:selector xpath="rendeles"/>
    <xs:field xpath="@vid_f"/>
</xs:keyref>

<xs:keyref refer="termekKulcs" name="raktarTermekIdegenKulcs">
    <xs:selector xpath="nyilvantartas"/>
    <xs:field xpath="@tid_f"/>
</xs:keyref>

<xs:keyref refer="futarcegKulcs" name="rendelesFutarcegIdegenKulcs">
    <xs:selector xpath="futar"/>
    <xs:field xpath="@fid_f"/>
</xs:keyref>

<!-- Saját típusok -->
<xs:complexType name="vasarloTipus">
    <xs:sequence>
        <xs:element name="vezeteknev" type="xs:string"/>
        <xs:element name="keresztnev" type="xs:string"/>
        <xs:element name="cim">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="varos" type="xs:string"/>
                    <xs:element name="utca" type="xs:string"/>
                    <xs:element name="hazszam" type="xs:string"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="telefonszam" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="vid" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="termekTipus">
    <xs:sequence>
        <xs:element name="elnevezes" type="xs:string"/>
        <xs:element name="leiras" type="xs:string" minOccurs="0"/>
        <xs:element name="ar" type="xs:integer"/>
    </xs:sequence>
    <xs:attribute name="tid" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="futarcegTipus">
    <xs:sequence>
        <xs:element name="nev" type="xs:string"/>
        <xs:element name="cim" type="xs:string"/>
        <xs:element name="szallitasiido" type="xs:string"/>
        <xs:element name="ar" type="xs:integer"/>
    </xs:sequence>
    <xs:attribute name="fid" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="nyilvantartasTipus">
    <xs:sequence>
        <xs:element name="mennyiseg" type="xs:integer"/>
    </xs:sequence>
    <xs:attribute name="tid_f" type="xs:string" use="required"/>
</xs:complexType>
```

```
<xs:complexType name="rendelesTipus">
    <xs:sequence>
        <xs:element name="datum">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="ev" type="xs:string"/>
                    <xs:element name="honap" type="xs:string"/>
                    <xs:element name="nap" type="xs:string"/>
                    <xs:element name="ora" type="xs:string"/>
                    <xs:element name="perc" type="xs:string"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="tartalom">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="arucikk" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="tid_f" type="xs:string"/>
                                <xs:element name="mennyiseg" type="xs:integer"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="vegosszeg" type="xs:integer"/>
        <xs:element name="futar">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="fid_f" type="xs:string"/>
                    <xs:element name="felvetel" type="xs:string"/>
                    <xs:element name="kezbesites" type="xs:string"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="rid" type="xs:string" use="required"/>
    <xs:attribute name="vid_f" type="xs:string" use="required"/>
</xs:complexType>
```

2. feladat

Az egyedeknek megfelelően kialakítottam a megfelelő osztályokat, minden osztályban van egy statikus tömb, amiben az egyes példányokat tárolom. A beolvasáskor ezekbe a tömbökbe töltöm bele az adatokat.

a) Adatolvasás

Az XML fájlból beolvasom az adatokat és az adatoknak megfelelő osztályokból létrehozott statikus tömbbe teszem az egyes példányokat. Minden egyes egyed beolvasásáért külön metódus felel.

A vásárlók adatainak beolvasása

```
public static void ReadVasarlo() throws SAXException,
        IOException, ParserConfigurationException {
    File xmlFile = new File("XMLTRNA8A.xml");

    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = factory.newDocumentBuilder();

    Document doc = dBuilder.parse(xmlFile);

    doc.getDocumentElement().normalize();

    NodeList nList = doc.getElementsByTagName("vasarlo");
    Vasarlo.CreateArray(nList.getLength());
    for (int i = 0;i < nList.getLength();i++){
        Node nNode = nList.item(i);

        if (nNode.getNodeType() == Node.ELEMENT_NODE) {
            Element elem = (Element) nNode;
            String vid = elem.getAttribute("vid");

            Node node1 = elem.getElementsByTagName("vezeteknev").item(0);
            String veznev = node1.getTextContent();

            Node node2 = elem.getElementsByTagName("keresztnev").item(0);
            String kernev = node2.getTextContent();

            Node node3 = elem.getElementsByTagName("varos").item(0);
            String varos = node3.getTextContent();

            Node node4 = elem.getElementsByTagName("utca").item(0);
            String utca = node4.getTextContent();

            Node node5 = elem.getElementsByTagName("hazszam").item(0);
            String hazszam = node5.getTextContent();

            Node node6 = elem.getElementsByTagName("telefonszam").item(0);
            String telszam = node6.getTextContent();

            Vasarlo.AddToArray(new Vasarlo(vid, veznev, kernev, varos, utca, hazszam, telszam),i);
        }
    }
}
```

A termékek adatainak beolvasása

```
public static void ReadTermek() throws SAXException,
        IOException, ParserConfigurationException {
    File xmlFile = new File("XMLTRNA8A.xml");

    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = factory.newDocumentBuilder();

    Document doc = dBuilder.parse(xmlFile);

    doc.getDocumentElement().normalize();

    NodeList nList = doc.getElementsByTagName("termek");
    Termek.CreateArray(nList.getLength());
    for (int i = 0;i < nList.getLength();i++){
        Node nNode = nList.item(i);

        if (nNode.getNodeType() == Node.ELEMENT_NODE) {
            Element elem = (Element) nNode;
            String tid = elem.getAttribute("tid");

            Node node1 = elem.getElementsByTagName("elnevezes").item(0);
            String elnevezes = node1.getTextContent();

            String leiras = "-";
            if (elem.getChildNodes().item(3).getNodeName() == "leiras") {
                Node node2 = elem.getElementsByTagName("leiras").item(0);
                leiras = node2.getTextContent();
            }

            Node node3 = elem.getElementsByTagName("ar").item(0);
            String ar = node3.getTextContent();

            Termek.AddToArray(new Termek(tid, elnevezes, leiras, Integer.parseInt(ar)),i);
        }
    }
}
```

A futárcégek adatainak beolvasása

```
public static void ReadFutarceg() throws SAXException,
        IOException, ParserConfigurationException {
    File xmlFile = new File("XMLTRNA8A.xml");

    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = factory.newDocumentBuilder();

    Document doc = dBuilder.parse(xmlFile);

    doc.getDocumentElement().normalize();

    NodeList nList = doc.getElementsByTagName("futarceg");
    Futarceg.CreateArray(nList.getLength());
    for (int i = 0;i < nList.getLength();i++){
        Node nNode = nList.item(i);

        if (nNode.getNodeType() == Node.ELEMENT_NODE) {
            Element elem = (Element) nNode;
            String fid = elem.getAttribute("fid");

            Node node1 = elem.getElementsByTagName("nev").item(0);
            String nev = node1.getTextContent();

            Node node2 = elem.getElementsByTagName("cim").item(0);
            String cim = node2.getTextContent();

            Node node3 = elem.getElementsByTagName("szallitasiido").item(0);
            String szallitasiIdo = node3.getTextContent();

            Node node4 = elem.getElementsByTagName("ar").item(0);
            String ar = node4.getTextContent();


            Futarceg.AddToArray(new Futarceg(fid, nev, cim, szallitasiIdo, Integer.parseInt(ar)),i);
        }
    }
}
```

A nyilvántartás beolvasása

```
public static void ReadRaktar() throws SAXException,
        IOException, ParserConfigurationException {
    File xmlFile = new File("XMLTRNA8A.xml");

    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = factory.newDocumentBuilder();

    Document doc = dBuilder.parse(xmlFile);

    doc.getDocumentElement().normalize();

    NodeList nList = doc.getElementsByTagName("nyilvantartas");

    for (int i = 0;i < nList.getLength();i++){
        Node nNode = nList.item(i);

        if (nNode.getNodeType() == Node.ELEMENT_NODE) {
            Element elem = (Element) nNode;
            String tid = elem.getAttribute("tid_f");

            Node node1 = elem.getElementsByTagName("mennyiseg").item(0);
            String mennyiseg = node1.getTextContent();

            Termek.SetRaktarMennyiseg(Integer.parseInt(mennyiseg),tid);
        }
    }
}
```

A rendelések adatainak beolvasása

```
public static void ReadRendeles() throws SAXException,
        IOException, ParserConfigurationException {
    File xmlFile = new File("XMLTRNA8A.xml");

    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    DocumentBuilder dBuilder = factory.newDocumentBuilder();

    Document doc = dBuilder.parse(xmlFile);

    doc.getDocumentElement().normalize();

    NodeList nList = doc.getElementsByTagName("rendeles");
    Rendeles.CreateArray(nList.getLength());
    for (int i = 0;i < nList.getLength();i++){
        Node nNode = nList.item(i);

        if (nNode.getNodeType() == Node.ELEMENT_NODE) {
            Element elem = (Element) nNode;
            String rid = elem.getAttribute("rid");
            String vid = elem.getAttribute("vid_f");

            Node node1 = elem.getElementsByTagName("ev").item(0);
            String ev = node1.getTextContent();

            Node node2 = elem.getElementsByTagName("honap").item(0);
            String honap = node2.getTextContent();

            Node node3 = elem.getElementsByTagName("nap").item(0);
            String nap = node3.getTextContent();

            Node node4 = elem.getElementsByTagName("ora").item(0);
            String ora = node4.getTextContent();

            Node node5 = elem.getElementsByTagName("perc").item(0);
            String perc = node5.getTextContent();
```

```
            NodeList tList = elem.getElementsByTagName("arucikk");
            RendeltTermek[] rendeltTermekek = new RendeltTermek[tList.getLength()];
            for (int j = 0;j<tList.getLength();j++){
                Node tNode = tList.item(j);
                if (tNode.getNodeType() == Node.ELEMENT_NODE){
                    Element tElem = (Element) tNode;
                    Node node6 = tElem.getElementsByTagName("tid_f").item(0);
                    String tid = node6.getTextContent();

                    Node node7 = tElem.getElementsByTagName("mennyiseg").item(0);
                    String mennyiseg = node7.getTextContent();
                    rendeltTermekek[j] = new RendeltTermek(Termek.getTermekById(tid), Integer.parseInt(mennyiseg));
                }

            }
            /*
            Node node6 = elem.getElementsByTagName("tid_f").item(0);
            String tid = node6.getTextContent();

            Node node7 = elem.getElementsByTagName("mennyiseg").item(0);
            String mennyiseg = node7.getTextContent();
            */
            Node node8 = elem.getElementsByTagName("vegosszeg").item(0);
            String vegosszeg = node8.getTextContent();

            Node node9 = elem.getElementsByTagName("fid_f").item(0);
            String fid = node9.getTextContent();

            Node node10 = elem.getElementsByTagName("felvetel").item(0);
            String felvetel = node10.getTextContent();

            Node node11 = elem.getElementsByTagName("kezbesites").item(0);
            String kezbesites = node11.getTextContent();

            String datum = ev + "-" + honap + "-" + nap + "-" + ora + "-" + perc;

            Rendeles.AddToArray(new Rendeles(rid, vid, datum, rendeltTermekek, Integer.parseInt(vegosszeg), fid, felvetel, kezbesites),i);
        }
    }
```

b) Adatmódosítás

A Foxpost futárcég nevének átírása "Foxpost Kft."-re

```java
public static void FoxpostAtnevezese() throws ParserConfigurationException, IOException, SAXException, TransformerException {
    File file = new File("XMLTRNA8A.xml");

    DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
    Document document = documentBuilder.parse(file);

    document.getDocumentElement().normalize();

    NodeList NList = document.getElementsByTagName("futarceg");
    for (int i = 0; i < NList.getLength(); i++) {
        Node node = NList.item(i);

        if (node.getNodeType() == Node.ELEMENT_NODE) {

            NodeList childNodes = node.getChildNodes();

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

                Node childNode = childNodes.item(j);

                if ("nev".equals(childNode.getNodeName())) {

                    if ("FOXPOST".equals(childNode.getTextContent())) {
                        childNode.setTextContent("Foxpost Kft.");
                    }
                }
            }
        }
    }

    File newFile = new File("XMLTRNA8A.xml");

    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();

    DOMSource source = new DOMSource(document);

    System.out.println("Módosítás eredménye:");

    StreamResult consoleResult = new StreamResult(System.out);
    StreamResult resultModFile = new StreamResult(newFile);

    transformer.transform(source, consoleResult);
    transformer.transform(source, resultModFile );
}
```

Termék árának módosítása

```java
public static void TermekArModositas(String tid, String newAr) throws ParserConfigurationException, IOException, SAXException, TransformerException {
    File file = new File("XMLTRNA8A.xml");

    DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
    Document document = documentBuilder.parse(file);

    document.getDocumentElement().normalize();

    NodeList NList = document.getElementsByTagName("termek");
    for (int i = 0; i < NList.getLength(); i++) {
        Node node = NList.item(i);

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element elem = (Element) node;
            if (elem.getAttribute("tid").equals(tid)) {
                NodeList childNodes = node.getChildNodes();

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

                    Node childNode = childNodes.item(j);

                    if ("ar".equals(childNode.getNodeName())) {
                        childNode.setTextContent(newAr);
                    }
                }
            }
        }
    }
```

A vásárló telefonszámának módosítása

```java
public static void VasarloTelefonszamModositas(String vid, String newTelszam) throws ParserConfigurationException, IOException, SAXException, TransformerException {
    File file = new File("XMLTRNA8A.xml");

    DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
    Document document = documentBuilder.parse(file);

    document.getDocumentElement().normalize();

    NodeList NList = document.getElementsByTagName("vasarlo");
    for (int i = 0; i < NList.getLength(); i++) {
        Node node = NList.item(i);

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element elem = (Element) node;
            if (elem.getAttribute("vid").equals(vid)) {
                NodeList childNodes = node.getChildNodes();

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

                    Node childNode = childNodes.item(j);

                    if ("telefonszam".equals(childNode.getNodeName())) {
                        childNode.setTextContent(newTelszam);
                    }
                }
            }
        }
    }

    File newFile = new File("XMLTRNA8A.xml");

    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();

    DOMSource source = new DOMSource(document);

    System.out.println("Módosítás eredménye:");

    StreamResult consoleResult = new StreamResult(System.out);
    StreamResult resultModFile = new StreamResult(newFile);

    transformer.transform(source, consoleResult);
    transformer.transform(source, resultModFile );
}
```

Futarcég árának módosítása

```java
public static void FutarcegArModositas(String fid, String newAr) throws ParserConfigurationException, IOException, SAXException, TransformerException {
    File file = new File("XMLTRNA8A.xml");

    DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
    Document document = documentBuilder.parse(file);

    document.getDocumentElement().normalize();

    NodeList NList = document.getElementsByTagName("futarceg");
    for (int i = 0; i < NList.getLength(); i++) {
        Node node = NList.item(i);

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element elem = (Element) node;
            if (elem.getAttribute("fid").equals(fid)) {
                NodeList childNodes = node.getChildNodes();

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

                    Node childNode = childNodes.item(j);

                    if ("ar".equals(childNode.getNodeName())) {
                        childNode.setTextContent(newAr);
                    }
                }
            }
        }
    }

    File newFile = new File("XMLTRNA8A.xml");

    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();

    DOMSource source = new DOMSource(document);

    System.out.println("Módosítás eredménye:");

    StreamResult consoleResult = new StreamResult(System.out);
    StreamResult resultModFile = new StreamResult(newFile);

    transformer.transform(source, consoleResult);
    transformer.transform(source, resultModFile );
}
```

Raktár bővítése

```java
public static void RaktarBovites(String tid, int mennyiseg) throws ParserConfigurationException, IOException, SAXException, TransformerException {
    File file = new File("XMLTRNA8A.xml");

    DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
    Document document = documentBuilder.parse(file);

    document.getDocumentElement().normalize();

    NodeList NList = document.getElementsByTagName("nyilvantartas");
    for (int i = 0; i < NList.getLength(); i++) {
        Node node = NList.item(i);

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element elem = (Element) node;
            if (elem.getAttribute("tid_f").equals(tid)) {
                NodeList childNodes = node.getChildNodes();

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

                    Node childNode = childNodes.item(j);

                    if ("mennyiseg".equals(childNode.getNodeName())) {
                        childNode.setTextContent(String.valueOf(Integer.parseInt(childNode.getTextContent())+mennyiseg));
                    }
                }
            }
        }
    }

    File newFile = new File("XMLTRNA8A.xml");

    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();

    DOMSource source = new DOMSource(document);

    System.out.println("Módosítás eredménye:");

    StreamResult consoleResult = new StreamResult(System.out);
    StreamResult resultModFile = new StreamResult(newFile);

    transformer.transform(source, consoleResult);
    transformer.transform(source, resultModFile );
}
```

c) Adatlekérdezés

Ez a metódus lekérdezi a paraméterben átadott árnál olcsóbb termékek adatait.

```java
public static void QueryOlcsobbTermekMint(int input){
    try {
        File file = new File("XMLTRNA8A.xml");

        DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document document = documentBuilder.parse(file);

        document.getDocumentElement().normalize();

        System.out.println(input + " Ft-nál olcsóbb termékek: ");
        NodeList nList2 = document.getElementsByTagName("termek");

        for (int i = 0; i < nList2.getLength(); i++) {

            Node nNode = nList2.item(i);

            if (nNode.getNodeType()==Node.ELEMENT_NODE) {

                Element elem = (Element) nNode;
                NodeList childNodes = nNode.getChildNodes();

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

                    Node childNode = childNodes.item(j);

                    if ("ar".equals(childNode.getNodeName())) {

                        if (Integer.valueOf(childNode.getTextContent()) < input) {

                            String tid = elem.getAttribute("tid");
                            Node node1 = elem.getElementsByTagName("elnevezes").item(0);
                            String elnevezes = node1.getTextContent();
                            Node node2 = elem.getElementsByTagName("leiras").item(0);
                            String leiras = node2.getTextContent();
                            Node node3 = elem.getElementsByTagName("ar").item(0);
                            String ar = node3.getTextContent();

                            System.out.println("TID: " + tid);
                            System.out.println("Elnevezés: " + elnevezes);
                            System.out.println("Leírás: " + leiras);
                            System.out.println("Ár: " + ar);

                        }
                    }
                }

            }

        }
    } catch (IOException e) {
        e.printStackTrace();
```

Termék adatainak lekérdezése paraméterben átadott azonosító alapján

```
public static void QueryTermekAzonositoAlapjan(String input){
    try {
        File file = new File("XMLTRNA8A.xml");

        DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document document = documentBuilder.parse(file);

        document.getDocumentElement().normalize();

        System.out.println("A lekérdezett termék: ");
        NodeList nList = document.getElementsByTagName("termek");

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

            Node nNode = nList.item(i);

            if (nNode.getNodeType()==Node.ELEMENT_NODE) {

                Element elem = (Element) nNode;

                if (elem.getAttribute("tid").equals(input)) {

                    String tid = elem.getAttribute("tid");
                    Node node1 = elem.getElementsByTagName("elnevezes").item(0);
                    String elnevezes = node1.getTextContent();
                    Node node2 = elem.getElementsByTagName("leiras").item(0);
                    String leiras = node2.getTextContent();
                    Node node3 = elem.getElementsByTagName("ar").item(0);
                    String ar = node3.getTextContent();

                    System.out.println("TID: " + tid);
                    System.out.println("Elnevezés: " + elnevezes);
                    System.out.println("Leírás: " + leiras);
                    System.out.println("Ár: " + ar);

                }
            }

        }
    } catch (IOException e) {
        e.printStackTrace();
    } catch (ParserConfigurationException e) {
        e.printStackTrace();
    } catch (SAXException e) {
        e.printStackTrace();
    }
}
```

Vásárló adatainak lekérdezése paraméterben átadott város alapján

```java
public static void QueryVasarloVarosAlapjan(String input){
    try {
        File file = new File("XMLTRNA8A.xml");

        DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document document = documentBuilder.parse(file);

        document.getDocumentElement().normalize();

        System.out.println("A lekérdezett vásárló: ");
        NodeList nList = document.getElementsByTagName("vasarlo");

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

            Node nNode = nList.item(i);

            if (nNode.getNodeType()==Node.ELEMENT_NODE) {

                Element elem = (Element) nNode;
                Node node = elem.getElementsByTagName("varos").item(0);

                if (node.getTextContent().equals(input)) {

                    String vid = elem.getAttribute("vid");
                    Node node1 = elem.getElementsByTagName("vezeteknev").item(0);
                    String veznev = node1.getTextContent();
                    Node node2 = elem.getElementsByTagName("keresztnev").item(0);
                    String kernev = node2.getTextContent();
                    Node node3 = elem.getElementsByTagName("varos").item(0);
                    String varos = node3.getTextContent();
                    Node node4 = elem.getElementsByTagName("utca").item(0);
                    String utca = node4.getTextContent();
                    Node node5 = elem.getElementsByTagName("hazszam").item(0);
                    String hazszam = node5.getTextContent();
                    Node node6 = elem.getElementsByTagName("telefonszam").item(0);
                    String telszam = node6.getTextContent();

                    System.out.println("ID: " + vid);
                    System.out.println("Vezeteknev: " + veznev);
                    System.out.println("Keresztnev: " + kernev);
                    System.out.println("Varos: " + varos);
                    System.out.println("Utca: " + utca);
                    System.out.println("Hazszam: " + hazszam);
                    System.out.println("Telefonszam: " + telszam);

                }
            }

        }
    } catch (IOException e) {
        e.printStackTrace();
    } catch (ParserConfigurationException e) {
        e.printStackTrace();
    } catch (SAXException e) {
        e.printStackTrace();
    }
}
```

A 2022-ben történt rendelések lekérdezése

```java
public static void QueryRendelesek2022ben(){
    try {
        File file = new File("XMLTRNA8A.xml");

        DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document document = documentBuilder.parse(file);

        document.getDocumentElement().normalize();

        System.out.println("A lekérdezett rendelés(ek): ");
        NodeList nList = document.getElementsByTagName("rendeles");

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

            Node nNode = nList.item(i);

            if (nNode.getNodeType()==Node.ELEMENT_NODE) {

                Element elem = (Element) nNode;
                Node node = elem.getElementsByTagName("ev").item(0);

                if (node.getTextContent().equals("2022")) {

                    String rid = elem.getAttribute("rid");
                    String vid = elem.getAttribute("vid_f");
                    Node node1 = elem.getElementsByTagName("ev").item(0);
                    String ev = node1.getTextContent();
                    Node node2 = elem.getElementsByTagName("honap").item(0);
                    String honap = node2.getTextContent();
                    Node node3 = elem.getElementsByTagName("nap").item(0);
                    String nap = node3.getTextContent();
                    Node node4 = elem.getElementsByTagName("ora").item(0);
                    String ora = node4.getTextContent();
                    Node node5 = elem.getElementsByTagName("perc").item(0);
                    String perc = node5.getTextContent();

                    NodeList tList = elem.getElementsByTagName("arucikk");
                    RendeltTermek[] rendeltTermekek = new RendeltTermek[tList.getLength()];
                    for (int j = 0;j<tList.getLength();j++){
                        Node tNode = tList.item(j);
                        if (tNode.getNodeType() == Node.ELEMENT_NODE){
                            Element tElem = (Element) tNode;
                            Node node6 = tElem.getElementsByTagName("tid_f").item(0);
                            String tid = node6.getTextContent();
```

```
                    Node node7 = tElem.getElementsByTagName("mennyiseg").item(0);
                    String mennyiseg = node7.getTextContent();
                    rendeltTermekek[j] = new RendeltTermek(Termek.getTermekById(tid), Integer.parseInt(mennyiseg));
                }

            }
            Node node8 = elem.getElementsByTagName("vegosszeg").item(0);
            String vegosszeg = node8.getTextContent();

            Node node9 = elem.getElementsByTagName("fid_f").item(0);
            String fid = node9.getTextContent();

            Node node10 = elem.getElementsByTagName("felvetel").item(0);
            String felvetel = node10.getTextContent();

            Node node11 = elem.getElementsByTagName("kezbesites").item(0);
            String kezbesites = node11.getTextContent();

            System.out.println("RID: " + rid);
            System.out.println("VID: " + vid);
            System.out.println("Ev: " + ev);
            System.out.println("Honap: " + honap);
            System.out.println("Nap: " + nap);
            System.out.println("Ora: " +ora);
            System.out.println("Perc: " + perc);
            for (int j=0;j<rendeltTermekek.length;j++){
                System.out.println((j+1) + ". termek: ");
                System.out.println("\tTID:" + rendeltTermekek[j].getTid());
                System.out.println("\tMennyiseg:" + rendeltTermekek[j].getRendeltMennyiseg());
            }
            System.out.println("Vegosszeg: " + vegosszeg);
            System.out.println("FID: " + fid);
            System.out.println("Felvetel: " + felvetel);
            System.out.println("Kezbesites: " + kezbesites);
            System.out.println();

        }
    }

    }
} catch (IOException e) {
    e.printStackTrace();
} catch (ParserConfigurationException e) {
    e.printStackTrace();
} catch (SAXException e) {
    e.printStackTrace();
}
}
```