JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Pizzéria

Készítette: Ónodi Bence

Neptunkód: RYSNLC

Dátum: 2023.12.05

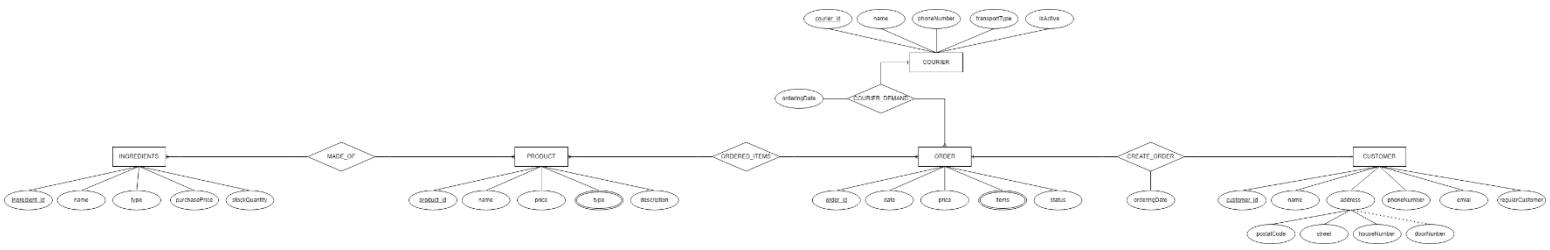
Tartalom

1. A feladat témája

A feladat témája egy étterem által kiszállításra kerülő rendeléseknek a nyilvántartása, melyben számon tudjuk tartani a futárokat, illetve azt is, hogy milyen fajta alapanyagok kellenek a rendelés elkészítéséhez.

- Ingredients egyed tulajdonságai
  - ingredient_id: az alapanyag egyedi azonosítója
  - name: az alapanyag neve
  - type: az alapanyag típusa pl.: feltét
  - purchasePrice: mennyiért lehet beszerezni az alapanyagot
  - stockQuantity: mennyi van az alapanyagból raktáron
- Product egyed tulajdonságai
  - product_id: az egyed egyedi azonosítója
  - name: a termék neve
  - price: a termék ára
  - type: többértékű tulajdonság ami a termék típusát adja meg
  - description: a termék leírása
- Order egyed tulajdonságai
  - order_id: a megrendelés egyedi kulcsa
  - date: a megrendelés ideje
  - price: a megrendelés végösszege
  - items: a megrendelt termékek, többértékű tulajdonság
  - status: a megrendelés állapota
- Courier egyed tulajdonságai
  - courier_id: a futár egyedi azonosítója
  - name: a futár neve
  - phoneNumber: a futár telefonszáma
  - transportType: milyen jármű segítségével szállít a futár
  - isActive: dolgozik e éppen a futár
- Customer egyed tulajdonságai
  - customer_id: a vevő kulcsa
  - name: a vevő neve
  - address: összetett tulajdonság, a vevő címe
  - phoneNumber: a vevő telefonszáma
  - email: a vevő email címe
  - regularCustomer: törzsvendég e

A feladat ER modellje:



Az egyedek között lévő kapcsolatok:

Product és Order közötti kapcsolat: Ordered_Items

- Több-több kapcsolat van közöttük, mivel egy termék többször is szerepelhet egy rendelésben, illetve egy rendeléshez több termék is tartozhat

Ingredients és Product közötti kapcsolat: Made_Of

- Több-több kapcsolat van közöttük, mivel egy alapanyagot több termékhez is fel lehet használni, illetve több alapanyagot is lehet egy termékhez.

Order és Courier közötti kapcsolat: Courier_Demand

- Egy-több kapcsolat van közöttük, mivel több rendelést is tud egy futár kiszállítani.

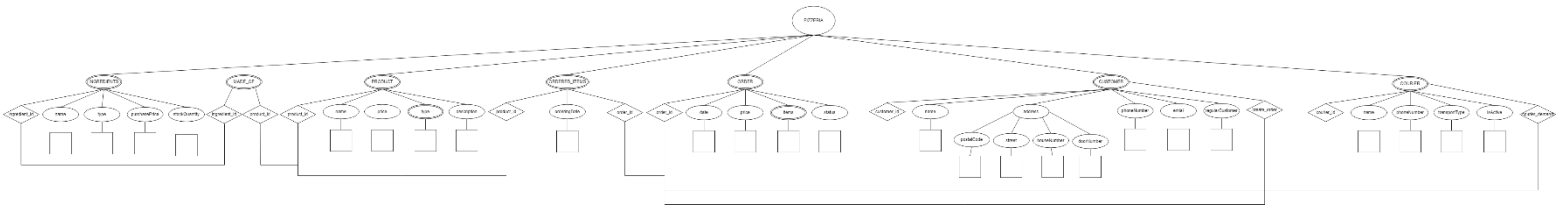
Order és Customer közötti kapcsolat: Create_Order

- Egy-egy kapcsolat van közöttük, mivel egy vevőhöz egy rendelés tartozhat csak, ami aktív, illetve egy megrendelés csak egy vevőhöz tartozik.

2. Az ER modell konvertálása XDM modellé

Az XDM modell alkalmazásakor háromféle jelölést használunk: ellipszist, rombuszt és téglalapot. Az ellipszis az elemeket jelképezi, amelyek minden egyes egyedből származnak, és a tulajdonságok is részükké válnak. A rombusz az attribútumokat ábrázolja, melyek a kulcs tulajdonságokból erednek. A téglalap pedig a szöveget jelképezi, amely az XML dokumentumban fog megjelenni. Azok az elemek, amelyek többször is előfordulhatnak, dupla ellipszissel vannak jelölve. Az idegenkulcsok és kulcsok közötti kapcsolatot pedig szaggatott vonalas nyíllal ábrázoljuk..

A feladat XDM modellje:



3. XML dokumentum készítése

A következő lépésben az XDM modell alapján elkészítettem az XML dokumentumot is. Először a root elementtel kezdtem, ami a Pizzéria elem volt. Mindezek után a többértékű gyerekelemeiből létrehoztam legalább 3-3 példányt is. Ezekhez megadtam az attribútumokat is, amik a kulcsok, idegenkulcsok. Ezeknek is létrehoztam a gyerekelemeit, amelyek között volt többértékű és összetett értékű elem is.

XML dokumentum forráskódja

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

```
<Pizzeria_RYSNLC        xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="XMLSchema_RYSNLC.xsd">

    <!--Ingredients-->

    <ingredients ingredient_id="1">
        <name>Flour</name>
        <type>Base</type>
        <purchasePrice>500</purchasePrice>
        <stockQuantity>1000</stockQuantity>
    </ingredients>

    <ingredients ingredient_id="2">
        <name>Sugar</name>
        <type>Sweetener</type>
        <purchasePrice>600</purchasePrice>
        <stockQuantity>800</stockQuantity>
    </ingredients>

    <ingredients ingredient_id="3">
        <name>Milk</name>
        <type>Dairy</type>
        <purchasePrice>300</purchasePrice>
        <stockQuantity>500</stockQuantity>
    </ingredients>

    <ingredients ingredient_id="4">
        <name>Tomatoes</name>
        <type>Vegetable</type>
        <purchasePrice>160</purchasePrice>
        <stockQuantity>600</stockQuantity>
    </ingredients>

    <ingredients ingredient_id="5">
        <name>Chicken</name>
        <type>Meat</type>
        <purchasePrice>1200</purchasePrice>
        <stockQuantity>300</stockQuantity>
    </ingredients>

    <ingredients ingredient_id="6">
        <name>Fish</name>
        <type>Fish</type>
        <purchasePrice>1600</purchasePrice>
        <stockQuantity>200</stockQuantity>
```

```
    </ingredients>

    <!--Products-->

    <product product_id="1">
        <name>Spaghetti Bolognese</name>
        <price>2400</price>
        <type>Pasta</type>
        <description>Classic spaghetti with Bolognese sauce.</description>
    </product>

    <product product_id="2">
        <name>Chicken Caesar Salad</name>
        <price>2200</price>
        <type>American</type>
        <type>Salad</type>
            <description>Healthy salad with grilled chicken and Caesar
dressing.</description>
    </product>

    <product product_id="3">
        <name>Sushi Combo</name>
        <price>3700</price>
        <type>Japanese</type>
        <type>Sushi</type>
               <description>Assorted sushi rolls and sashimi in a
combo.</description>
    </product>

    <!--Made of kapcsolat-->

    <made_of ingredient_id="1" product_id="1"></made_of>
    <made_of ingredient_id="4" product_id="1"></made_of>
    <made_of ingredient_id="5" product_id="1"></made_of>

    <made_of ingredient_id="5" product_id="2"></made_of>
    <made_of ingredient_id="4" product_id="2"></made_of>

    <made_of ingredient_id="6" product_id="3"></made_of>

    <!--Orders-->

    <order order_id="1">
        <date>2023.01.15</date>
        <price>4600</price>
        <item>Spaghetti Bolognese</item>
```

```
    <item>Chicken Caesar Salad</item>
    <status>Completed</status>
</order>

<order order_id="2">
    <date>2023.02.02</date>
    <price>2200</price>
    <item>Chicken Caesar Salad</item>
    <status>Processing</status>
</order>

<order order_id="3">
    <date>2023.03.10</date>
    <price>5600</price>
    <item>Sushi Combo</item>
    <item>Miso Soup</item>
    <status>Shipping</status>
</order>

<!--Order items kapcsolat-->

<ordered_items product_id="1" order_id="1">
    <orderingDate>2023.01.15</orderingDate>
</ordered_items>

<ordered_items product_id="1" order_id="1">
    <orderingDate>2023.02.02</orderingDate>
</ordered_items>

<ordered_items product_id="1" order_id="1">
    <orderingDate>2023.03.10</orderingDate>
</ordered_items>

<!--Customers-->

<customer customer_id="1" create_order="1">
    <name>John Doe</name>
    <address>
        <postalcode>12345</postalcode>
        <street>Main Street</street>
        <houseNumber>10</houseNumber>
        <doorNumber>2</doorNumber>
    </address>
    <phoneNumber>06304565434</phoneNumber>
    <email>john.doe@example.com</email>
    <regularCustomer>true</regularCustomer>
```

```
</customer>

<customer customer_id="2" create_order="2">
    <name>Alice Smith</name>
    <address>
        <postalcode>54321</postalcode>
        <street>Maple Avenue</street>
        <houseNumber>5</houseNumber>
        <doorNumber>1</doorNumber>
    </address>
    <phoneNumber>06704568978</phoneNumber>
    <email>alice.smith@example.com</email>
    <regularCustomer>false</regularCustomer>
</customer>

<customer customer_id="3" create_order="3">
    <name>Bob Johnson</name>
    <address>
        <postalcode>67890</postalcode>
        <street>Pine Street</street>
        <houseNumber>8</houseNumber>
        <doorNumber>3</doorNumber>
    </address>
    <phoneNumber>06304239825</phoneNumber>
    <email>bob.johnson@example.com</email>
    <regularCustomer>true</regularCustomer>
</customer>

<!--Couriers-->

<courier courier_id="1" courier_demand="1">
    <name>Mike Johnson</name>
    <phoneNumber>+15551234567</phoneNumber>
    <transportType>Bicycle</transportType>
    <isActive>true</isActive>
</courier>

<courier courier_id="2" courier_demand="2">
    <name>Sara Davis</name>
    <phoneNumber>+18882345678</phoneNumber>
    <transportType>Car</transportType>
    <isActive>true</isActive>
</courier>

<courier courier_id="3" courier_demand="3">
    <name>Chris Martinez</name>
```

```
        <phoneNumber>+16667778888</phoneNumber>
        <transportType>Motorcycle</transportType>
        <isActive>true</isActive>
    </courier>

</Pizzeria_RYSNLC>
```

4. XMLSchema készítése

Az XML dokumentum validációjához létre kellett hoznom egy sémát, az XMLSchemat. Ebben először létrehoztam az egyszerű elemeket, amelyekre később tudok referálni, illetve az ezekhez tartozó saját típusokat is. Itt használtam enumerationt is illetőlegesen regexeket is. Ezek után hoztam létre a tényleges felépítést, amelyben a korábban említettek szerint referáltam az előre létrehozott egyszerű elemekre, illetve beállítottam a minimum és maximum előfordulást is, hiszen a többértékű elemeknél ez szükséges. Megadtam az attribútumokat is, meg legvégül a kulcsokat, idegenkulcsokat, és 1:1 kapcsolatokat is.

Az XMLSchema kódja

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

    <!--Egyszerű elemek-->

    <xs:element name="name" type="xs:string" />
    <xs:element name="type" type="ingredients_food_type" />
    <xs:element name="purchasePrice" type="xs:int" />
    <xs:element name="stockQuantity" type="xs:int" />

    <xs:element name="price" type="xs:int" />
    <xs:element name="description" type="xs:string" />

    <xs:element name="date" type="idoTipus" />
    <xs:element name="item" type="xs:string" />
    <xs:element name="status" type="status_type" />

    <xs:element name="orderingDate" type="idoTipus" />

    <xs:element name="postalcode" type="xs:int" />
    <xs:element name="street" type="xs:string" />
    <xs:element name="houseNumber" type="xs:int" />
    <xs:element name="doorNumber" type="xs:int" />
    <xs:element name="email" type="emailTipus" />
    <xs:element name="phoneNumber" type="telefonszamTipus" />
```

```
    <xs:element name="transportType" type="xs:string" />
    <xs:element name="isActive" type="xs:string" />

    <!--Saját típusok-->

    <xs:simpleType name="ingredients_food_type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Base" />
            <xs:enumeration value="Sweetener" />
            <xs:enumeration value="Dairy" />
            <xs:enumeration value="Vegetable" />
            <xs:enumeration value="Meat" />
            <xs:enumeration value="Fish" />
            <xs:enumeration value="Pata" />
            <xs:enumeration value="Italian" />
            <xs:enumeration value="Salad" />
            <xs:enumeration value="American" />
            <xs:enumeration value="Japanese" />
            <xs:enumeration value="Sushi" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="status_type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Shipping" />
            <xs:enumeration value="Processing" />
            <xs:enumeration value="Completed" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="telefonszamTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="(06(20|30|31|50|60|70)\d{7})"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="idoTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="([12]\d{3}.(0[1-9]|1[0-2]).(0[1-
9]|[12]\d|3[01]))" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="emailTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="[\w\.]+@([\w]+\.)+[\w]{2,4}" />
```

```
        </xs:restriction>
    </xs:simpleType>

    <!--Felépítés-->

    <xs:element name="Pizzeria_RYSNLC">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ingredients" minOccurs="1"
maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="name" />
                            <xs:element name="type" />
                            <xs:element ref="purchasePrice" />
                            <xs:element ref="stockQuantity" />
                        </xs:sequence>
                        <xs:attribute name="ingredient_id" type="xs:int" />
                    </xs:complexType>
                </xs:element>
                <xs:element name="product" minOccurs="1"
maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="name" />
                            <xs:element ref="price" />
                            <xs:element ref="type" minOccurs="1"
maxOccurs="unbounded"/>
                            <xs:element ref="description" />
                        </xs:sequence>
                        <xs:attribute name="product_id" type="xs:int" />
                    </xs:complexType>
                </xs:element>
                <xs:element name="made_of" minOccurs="1"
maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:attribute name="ingredient_id" type="xs:int" />
                        <xs:attribute name="product_id" type="xs:int" />
                    </xs:complexType>
                </xs:element>
                <xs:element name="order" minOccurs="1"
maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="date" />
                            <xs:element ref="price" />
```

```
                            <xs:element ref="item" minOccurs="1"
maxOccurs="unbounded" />
                            <xs:element ref="status" />
                        </xs:sequence>
                        <xs:attribute name="order_id" type="xs:int" />
                    </xs:complexType>
                </xs:element>
                <xs:element name="ordered_items" minOccurs="1"
maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="orderingDate" />
                        </xs:sequence>
                        <xs:attribute name="product_id" type="xs:int" />
                        <xs:attribute name="order_id" type="xs:int" />
                    </xs:complexType>
                </xs:element>
                <xs:element name="customer" minOccurs="1"
maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="name" />
                            <xs:element name="address" minOccurs="1"
maxOccurs="1" >
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element ref="postalcode" />
                                        <xs:element ref="street" />
                                        <xs:element ref="houseNumber" />
                                        <xs:element ref="doorNumber" />
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="phoneNumber" />
                            <xs:element name="email" />
                            <xs:element name="regularCustomer" />
                        </xs:sequence>
                        <xs:attribute name="customer_id" type="xs:int" />
                        <xs:attribute name="create_order" type="xs:int" />
                    </xs:complexType>
                </xs:element>
                <xs:element name="courier" minOccurs="1"
maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="name" />
```

```
                            <xs:element name="phoneNumber" />
                            <xs:element name="transportType" />
                            <xs:element name="isActive" />
                        </xs:sequence>
                        <xs:attribute name="courier_id" type="xs:int" />
                        <xs:attribute name="courier_demand" type="xs:int"
/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>

        <!--Kulcsok-->

        <xs:key name="ingredient_kulcs">
            <xs:selector xpath="ingredients" />
            <xs:field xpath="@ingredient_id" />
        </xs:key>

        <xs:key name="product_kulcs">
            <xs:selector xpath="product" />
            <xs:field xpath="@product_id" />
        </xs:key>

        <xs:key name="order_kulcs">
            <xs:selector xpath="order" />
            <xs:field xpath="@order_id" />
        </xs:key>

        <xs:key name="customer_kulcs">
            <xs:selector xpath="customer" />
            <xs:field xpath="@customer_id" />
        </xs:key>

        <xs:key name="courier_kulcs">
            <xs:selector xpath="courier" />
            <xs:field xpath="@courier_id" />
        </xs:key>

        <!--Idegen kulcsok-->

        <xs:keyref refer="ingredient_kulcs"
name="made_of_ingredient_idegen_kulcs">
            <xs:selector xpath="made_of" />
            <xs:field xpath="@ingredient_id" />
        </xs:keyref>
```

```
        <xs:keyref refer="product_kulcs"
name="made_of_product_idegen_kulcs">
            <xs:selector xpath="made_of" />
            <xs:field xpath="@product_id" />
        </xs:keyref>

        <xs:keyref refer="order_kulcs" name="ordered_items_idegen_kulcs">
            <xs:selector xpath="ordered_items" />
            <xs:field xpath="@order_id" />
        </xs:keyref>

        <xs:keyref refer="order_kulcs" name="create_order_idegen_kulcs">
            <xs:selector xpath="customer" />
            <xs:field xpath="@create_order" />
        </xs:keyref>

        <xs:keyref refer="order_kulcs" name="courier_demand_idegen_kulcs">
            <xs:selector xpath="courier" />
            <xs:field xpath="@courier_demand" />
        </xs:keyref>

        <!--1:1-->

        <xs:unique name="create_order">
            <xs:selector xpath="customer" />
            <xs:field xpath="@create_order" />
        </xs:unique>

    </xs:element>

</xs:schema>
```

5. DOM adatolvasás

Ennek a feladatnak az elkezdésekor először is azt kellett beállítanom, hogy melyik fájl az, amiből a beolvasást el kell végezni. Ezt a fájlt a projekt mappájában helyeztem el. Létrehoztam egy document elemet, aminek a segítségével le tudtam kérdezni a root elementet, illetve mindezek után a parent, illetve gyemekelemeket is. Létre kell hozni NodeList-et is, amellyel a többszöri előfordulású elemeket tudom eltárolni. Miután ez megtörtént ezeken végig kell iterálni ahhoz, hogy a különböző attribútumokat, illetve gyermekelemeket ki tudjam belőle nyerni, majd kiíratni ezeket. Ezekben szintén lehet olyan, amelyekben többértékű elem van, itt megvizsgálom azt, hogy itt ezekből több elem van e, vagy csak egy. Amennyiben igen, itt is for ciklussal iterálok, és kiíratom őket.
Kód:

```
package DOMParse_RYSNLC;


import java.io.File;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class DOMReadRYSNLC {

    public static void main(String[] args) {
        try {

            String xmlFilePath = "1feladat\\XML_RYSNLC.xml";

            // XML dokumentum létrehozása és beolvasása
            File xmlFile = new File(xmlFilePath);
            DocumentBuilderFactory                dbFactory                =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);

            // Dokumentum elemeinek kiírása a konzolra
            printNode(doc.getDocumentElement(), "");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // Rekurzív metódus az XML elemek kiírására
    private static void printNode(Node node, String behúzás) {

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

            //elemek előtt enter
            if (node.getParentNode().getNodeType() == Node.ELEMENT_NODE) {

                System.out.println("");

            }

            //tagek kiírása
            System.out.println(behúzás + "<" + node.getNodeName() + ">");

            //tagnév kiírása
            NodeList nodeList = node.getChildNodes();
            for (int i = 0; i < nodeList.getLength(); i++) {
                printNode(nodeList.item(i), behúzás + "  ");
            }

            //tagek lezárása
            System.out.println(behúzás + "</" + node.getNodeName() + ">");

            //tag után üres sor
            System.out.println("");

        } else if (node.getNodeType() == Node.TEXT_NODE) {
```

```
            //tag szöveg tartalmának kiírása
            String text = node.getNodeValue().trim();
            if (!text.isEmpty()) {
                System.out.println(behúzás + text);
            }

        } else if (node.getNodeType() == Node.COMMENT_NODE) {

            //comment kiírása
            System.out.println("");
            System.out.println(behúzás + "<!-- " + node.getNodeValue() + " -
->");

        }
    }
}
```

6. DOM adatmódosítás

Az első ordernek a kulcsát választottam ki módosításra. Ez a kulcs attribútum több helyen is megjelenik mint például a customer, courier elemeknél is, ezért ezeket ott is megváltoztattam, illetve a különböző kapcsolati elemeknél is. A módosított fájlt kiírattam a konzolra, illetve egy új fájlba mentem.
Kód:

```
package DOMParse_RYSNLC;

import java.io.File;
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DOMModifyRYSNLC {

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

```
        try {

                                            File      inputFile      =      new
File("LEADNI\\2feladat\\DOMParse_RYSNLC\\XML_RYSNLC_2.xml.xml");

                          DocumentBuilderFactory   documentBuilderFactory   =
DocumentBuilderFactory.newInstance();
                                      DocumentBuilder     documentBuilder     =
documentBuilderFactory.newDocumentBuilder();

            Document doc = documentBuilder.parse(inputFile);

            // order attribútumának módosítása
            Node csoport = doc.getElementsByTagName("order").item(0);

            NamedNodeMap attr = csoport.getAttributes();
            Node nodeAttr = attr.getNamedItem("order_id");
            nodeAttr.setTextContent("4");

            // order item módosítás
            NodeList kszList = doc.getElementsByTagName("ordered_items");

            for (int i = 0; i < kszList.getLength(); i++) {
                Node node = kszList.item(i);

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

                    Element eElement = (Element) node;
                    String tagValue = eElement.getAttribute("order_id");

                    if ("1".equals(tagValue)) {
                        eElement.setAttribute("order_id", "4");
                    }

                }
            }

            // customer items módosítás
            NodeList cList = doc.getElementsByTagName("customer");

            for (int i = 0; i < cList.getLength(); i++) {
                Node node = cList.item(i);

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

                    Element eElement = (Element) node;
```

```
                    String tagValue = eElement.getAttribute("create_order");

                    if ("1".equals(tagValue)) {
                        eElement.setAttribute("create_order", "4");
                    }

                }
            }

            // courier item módosítás
            NodeList courList = doc.getElementsByTagName("courier");

            for (int i = 0; i < courList.getLength(); i++) {
                Node node = courList.item(i);

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

                    Element eElement = (Element) node;
                  String tagValue = eElement.getAttribute("courier_demand");

                    if ("1".equals(tagValue)) {
                        eElement.setAttribute("courier_demand", "4");
                    }

                }
            }

            // Tartalom írása
                                TransformerFactory    transformerFactory    =
TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();

            DOMSource source = new DOMSource(doc);

            System.out.println("-Módosított fájl-");

            StreamResult consoleResult = new StreamResult(System.out);
            StreamResult file = new StreamResult(inputFile);

            transformer.transform(source, consoleResult);
            transformer.transform(source, file);

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

```
}
```

7. DOM adatlekérdezés

A lekérdezéseket az XPath segítségével hajtottam végre, ebből 7-et csináltam, majd ezeket teszteltem, a kritériumoknak megfelelőket kiírattam a konzolra.

- a Pizzeria_RYSNLC root elem alapanyag gyerekelemei
- alapanyagok, amiknek van attribútuma
- alapanyagok, amik mint 500Ft-ba kerülnek
- alapanyagok típusa, amik több, mint 200Ft-ba kerülnek
- termékek első két eleme
- 2-es azonositójú termék
- a harmadik termék kiválasztása

Kód:

```
package DOMParse_RYSNLC;

import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DOMQueryRYSNLC {

    public static void main(String[] args) {

        try {

            // DocumentBuilder
                        DocumentBuilderFactory   documentBuilderFactory   =
DocumentBuilderFactory.newInstance();
                                    DocumentBuilder     documentBuilder     =
documentBuilderFactory.newDocumentBuilder();

                                                Document       document       =
documentBuilder.parse("LEADNI\\1feladat\\XML_RYSNLC.xml.xml");

            document.getDocumentElement().normalize();

            // XPath
            XPath xPath = XPathFactory.newInstance().newXPath();

            // a Pizzeria_RYSNLC root elem alapanyag gyerekelemei
```

```
            String expression = "Pizzeria_RYSNLC / ingredients";

            // alapanyagok, amiknek van attribútuma
            // String expression = "//ingredients[@*]";

            // alapanyagok, amik mint 500ba kerülnek
            // String expression = "//ingredients[purchasePrice='500']";

            // alapanyagok típusa, amik több, mint 200Ft-ba kerülnek
            // String expression = "//ingredients[purchasePrice>200]/type";

            // termékek első két eleme
            // String expression = "//product[position()<3]";

            // 2-es azonositójú termék
            // String expression = "//product[@product_id='2']";

            // a harmadik termék kiválasztáas
            // String expression = "Pizzeria_RYSNLC/product[3]";

           // keszitunk egy listat, majd az xPath kifejezest le kell forditani
es ki kell ertekelni
                                    NodeList     nodeList     =     (NodeList)
xPath.compile(expression).evaluate(document, XPathConstants.NODESET);

            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);
               System.out.println("\nAktualis elem: " + node.getNodeName());

                           if   (node.getNodeType()   ==   Node.ELEMENT_NODE   &&
node.getNodeName().equals("ingredients")) {

                    Element elem = (Element) node;

                        String id = elem.getAttribute("ingredient_id");

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

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

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

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

                        System.out.println("Alapanyag id-je: " + id);
                        System.out.println("Név: " + name);
                        System.out.println("Típus: " + type);
                        System.out.println("PurchasePrice: " + purchaseP);
                        System.out.println("StockQuantity: " + stockQ);

                }

                // alapanyagok típusa
                           if   (node.getNodeType()   ==   Node.ELEMENT_NODE   &&
node.getNodeName().equals("type")) {
```

```
                    Element element = (Element) node;

                                System.out.println("Alapanyag  típusa:  "  +
element.getTextContent());

                }

                //product kiiratasa
                            if  (node.getNodeType()  ==  Node.ELEMENT_NODE  &&
node.getNodeName().equals("product")) {

                    Element element = (Element) node;

                                            System.out.println("ID:    "    +
element.getAttribute("product_id"));

                    System.out.println(
                                                        "Termék   neve:   "   +
element.getElementsByTagName("name").item(0).getTextContent());

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

                            if (nodeList.item(i).getChildNodes().getLength()
> 3) {
                                int db = 0;
                                                          Node   node4   =
element.getElementsByTagName("type").item(0);
                                while (node4 != null) {
                                                                   node4   =
element.getElementsByTagName("type").item(db);
                                    if (node4 != null) {
                                       String type = node4.getTextContent();
                                           System.out.println("A típusa: " +
type);
                                    }
                                    db++;
                                }

                             }

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




                }
            }

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

```
        }
    }
}
```

8. DOM adatírás

Minden, az XML-ben szereplő gyerekelemhez létre kellett hoznom egy metódust, aminek a segítségével a root elemhez ezeket hozzá tudom adni őket. Ezekben a tagek-nek a nevét és a hozzájuk tartozó értékeket a paraméterek segítségével tudtam megadni, majd mindezek után a konzolra, illetve külön fájlba is kiíratom ezeket. Azt, hogy ezt rendben sikerült e végrehajtanom tudtam ellenőrizni az XMLSchema segítségével.

Kód:

```
package DOMParse_RYSNLC;

import java.io.File;

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 org.w3c.dom.Comment;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;

public class DOMWriteRYSNLC {

      public static void main(String argv[]) throws Exception {

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

            Document doc = dBuilder.newDocument();

            Element root = doc.createElementNS("RYSNLC", "Pizzeria_RYSNLC");
            doc.appendChild(root);

            // ingredients

            root.appendChild(createIngredients(doc,  "1",  "Flour",  "Base",
"500", "1000"));
            root.appendChild(createIngredients(doc,      "2",      "Sugar",
"Sweetener", "600", "800"));
            root.appendChild(createIngredients(doc,  "3",  "Milk",  "Dairy",
"300", "500"));

            Element             element             =             (Element)
doc.getElementsByTagName("ingredients").item(0);
```

```
            Comment comment = doc.createComment("Ingredients");
            element.getParentNode().insertBefore(comment, element);

            // products

            String[] t = {"pasta", "italian"};
        root.appendChild(createProduct(doc,  "1",  "Spaghetti",  "2400",  t,
"Classic spaghetti with Bolognese sauce."));
        String[] t2 = {"salad"};
        root.appendChild(createProduct(doc, "2", "Ceasar salad", "2200", t2,
"Healthy salad with grilled chicken and Caesar dressing."));
        String[] t3 = {"japanese", "sushi"};
        root.appendChild(createProduct(doc,  "3",  "Sushi  box",  "3600",  t3,
"Assorted sushi rolls and sashimi in a combo."));

            element = (Element) doc.getElementsByTagName("product").item(0);
            comment = doc.createComment("Products");
            element.getParentNode().insertBefore(comment, element);

            // made_of

            root.appendChild(createMadeOf(doc, "1", "1"));
            root.appendChild(createMadeOf(doc, "2", "2"));
            root.appendChild(createMadeOf(doc, "3", "3"));

            element = (Element) doc.getElementsByTagName("made_of").item(0);
            comment = doc.createComment("Made of kapcsolat");
            element.getParentNode().insertBefore(comment, element);

            // orders

            String[] i = {"Spaghetti Bolognese", "Ceasar salad"};
            root.appendChild(createOrder(doc,  "1",  "2023.01.15",  "4600",
i,"Completed"));

            String[] i2 = {"Sushi box"};
            root.appendChild(createOrder(doc,  "2",  "2023.06.15",  "3600",
i2,"Processing"));

            String[] i3 = {"Spaghetti Bolognese"};
            root.appendChild(createOrder(doc,  "3",  "2023.07.15",  "2400",
i3,"Shipping"));

            element = (Element) doc.getElementsByTagName("order").item(0);
            comment = doc.createComment("Orders");
            element.getParentNode().insertBefore(comment, element);

            // ordered_items

            root.appendChild(createOrderedItems(doc,        "1",        "1",
"2023.06.12"));
            root.appendChild(createOrderedItems(doc,        "2",        "2",
"2023.08.12"));
            root.appendChild(createOrderedItems(doc,        "3",        "3",
"2023.09.12"));

            element                        =                        (Element)
doc.getElementsByTagName("ordered_items").item(0);
            comment = doc.createComment("Order items kapcsolat");
            element.getParentNode().insertBefore(comment, element);
```

```
            // customer

        root.appendChild(createCustomer(doc, "1", "1", "John Doe", "12345",
"Main street", "10", "2", "06304565434", "john.doe@example.com", "true"));
        root.appendChild(createCustomer(doc,  "2",  "2",  "Alice  Smith",
"12345", "Maple street", "1", "75", "06304565434", "alice.doe@example.com",
"false"));
        root.appendChild(createCustomer(doc,  "3",  "3",  "Bob  Johnson",
"12345", "Pine street", "10", "24", "06304566666", "bob.johnson@example.com",
"true"));

            element                         =                         (Element)
doc.getElementsByTagName("customer").item(0);
            comment = doc.createComment("Customers");
            element.getParentNode().insertBefore(comment, element);

            // courier

            root.appendChild(createCourier(doc,  "1",  "1",  "Mike  Johnson",
"06705556665", "Car", "true" ));
            root.appendChild(createCourier(doc,  "2",  "2",  "Sara  Davis",
"06705557777", "Car", "true" ));
            root.appendChild(createCourier(doc, "3", "3", "Chris Martinez",
"06705558888", "Motorbike", "true" ));

            element = (Element) doc.getElementsByTagName("courier").item(0);
            comment = doc.createComment("Couriers");
            element.getParentNode().insertBefore(comment, element);

            // Transform

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

            transf.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
            transf.setOutputProperty(OutputKeys.INDENT, "yes");
            transf.setOutputProperty("{https://xml.apache.org/xslt}indent-
amount", "2");

            // File letrehozas

            DOMSource source = new DOMSource(doc);
            File myFile = new File("RYSNLC_write.xml");

            StreamResult console = new StreamResult(System.out);
            StreamResult file = new StreamResult(myFile);

            transf.transform(source, console);
            transf.transform(source, file);

      }

      private   static   Node   createIngredients(Document   doc,   String
ingredient_id, String name, String type, String purchasePrice,
                  String stockQuantity) {

            Element ing = doc.createElement("ingredients");

            ing.setAttribute("ingredient_id", ingredient_id);
            ing.appendChild(createElement(doc, "name", name));
```

```
            ing.appendChild(createElement(doc, "type", type));
            ing.appendChild(createElement(doc,              "purchasePrice",
purchasePrice));
            ing.appendChild(createElement(doc,              "stockQuantity",
stockQuantity));

            return ing;

      }

      private static Node createProduct(Document doc, String product_id,
String name,
                  String price, String[] type, String description) {

            Element pr = doc.createElement("product");

            pr.setAttribute("product_id", product_id);
            pr.appendChild(createElement(doc, "name", name));
            pr.appendChild(createElement(doc, "price", price));

            Node[] node = appendArray(doc, "type", type);

            for (int i = 0; i < type.length; i++) {
                  pr.appendChild(node[i]);
            }

            pr.appendChild(createElement(doc, "description", description));

            return pr;

      }

      private static Node createMadeOf(Document doc, String ingredient_id,
String product_id) {

            Element mo = doc.createElement("made_of");

            mo.setAttribute("ingredient_id", ingredient_id);
            mo.setAttribute("product_id", product_id);

            return mo;

      }

      private static Node createOrder(Document doc, String order_id, String
date,
        String price, String[] item, String status) {

            Element or = doc.createElement("order");

            or.setAttribute("order_id", order_id);
        or.appendChild(createElement(doc, "date", date));
            or.appendChild(createElement(doc, "price", price));

            Node[] node = appendArray(doc, "item", item);

            for (int i = 0; i < item.length; i++) {
                  or.appendChild(node[i]);
            }

            or.appendChild(createElement(doc, "status", status));
```

```
            return or;

      }

      private static Node createOrderedItems(Document doc, String order_id,
String product_id, String orderingDate) {

            Element oi = doc.createElement("ordered_items");

            oi.setAttribute("order_id", order_id);
            oi.setAttribute("product_id", product_id);
            oi.appendChild(createElement(doc,                "orderingDate",
orderingDate));

            return oi;

      }

      private static Node createCustomer(Document doc, String customer_id,
String create_order, String name,
                  String postalcode, String street, String houseNumber,
String doorNumber, String phoneNumber, String email,
                  String regularCustomer) {

            Element cElement = doc.createElement("customer");

            cElement.setAttribute("customer_id", customer_id);
            cElement.setAttribute("create_order", create_order);
            cElement.appendChild(createElement(doc, "name", name));

            Element cim = doc.createElement("address");
            cim.appendChild(createElement(doc, "postalcode", postalcode));
        cim.appendChild(createElement(doc, "street", street));
            cim.appendChild(createElement(doc, "houseNumber", houseNumber));
            cim.appendChild(createElement(doc, "doorNumber", doorNumber));

        cElement.appendChild(cim);

            cElement.appendChild(createElement(doc,           "phoneNumber",
phoneNumber));
            cElement.appendChild(createElement(doc, "email", email));
            cElement.appendChild(createElement(doc,       "regularCustomer",
regularCustomer));

            return cElement;

      }

      private static Node createCourier(Document doc, String courier_id,
String courier_demand,
      String name, String phoneNumber, String transportType, String isActive)
{

            Element c = doc.createElement("courier");

            c.setAttribute("courier_id", courier_id);
            c.setAttribute("courier_demand", courier_demand);
            c.appendChild(createElement(doc, "name", name));
            c.appendChild(createElement(doc, "phoneNumber", phoneNumber));
```

```
            c.appendChild(createElement(doc,                "transportType",
transportType));
            c.appendChild(createElement(doc, "isActive", isActive));

            return c;

      }

      private static Node createElement(Document doc, String name, String
value) {

            Element node = doc.createElement(name);
            node.appendChild(doc.createTextNode(value));

            return node;

      }

      private static Node[] appendArray(Document doc, String name, String[]
value) {

            Element nodes[] = new Element[value.length];

            for (int i = 0; i < value.length; i++) {

                  nodes[i] = doc.createElement(name);
                  nodes[i].appendChild(doc.createTextNode(value[i]));

            }

            return nodes;

      }

}
```