JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Pizzázó

Készítette: **Drig Dávid**
Neptunkód: **EZ3YRC**
Dátum: 2023.12.02

Tartalomjegyzék:

1a) A feladat témája

A beadandó témája egy olyan adatbázis, amely több pizzázót kezel. Rákereshetünk benne a pizzázóban dolgozó futárokra, vagy beszállítókra, a vevő adatait is lekérdezhetjük.

<u>Az ER modell egyedei és tulajdonságai</u>:

- **A Bankkártya egyed tulajdonságai**
  - Kártyaszám: A Bankkártya egyed elsődleges kulcsa.
  - Bank: A bank neve, amelyhez a bankkártya tartozik.
  - Lejárati dátum: A kártya lejárati dátuma.
  - Típus: A bankkártya típusa.

- **A Vevő egyed tulajdonságai**
  - VevőID: A Vevő egyed elsődleges kulcsa.
  - Név: A vevő neve.
  - Telefonszám: A vevő telefonszáma.
  - Cím: Összetett tulajdonság. A vevő címe.

- **A Pizza egyed tulajdonságai**
  - PizzaID: A Pizza egyed elsődleges kulcsa.
  - Teljes ár: A rendelt pizza/pizzák teljes ára. Származtatott tulajdonság.
  - Pizza neve: A pizza neve.
  - Méret: Többértékű tulajdonság. A pizza méretét tárolja.
  - Feltét: Többértékű tulajdonság. A pizzán lévő feltéteket tárolja.

- **A Futár egyed tulajdonságai**
  - FutárID: A Futár egyed elsődleges kulcsa.
  - Telefonszám: A futár telefonszáma.
  - Név: A futár neve.

- **A Beszállító egyed tulajdonságai**
  - BeszállítóID: A Beszállító egyed elsődleges kulcsa.
  - Elérhetőség: A beszállító elérhetősége.
  - Név: A beszállító cég neve.
  - Cím: Összetett tulajdonság. A beszállító cég címe.

- **A Pizzázó egyed tulajdonságai**
  - PizzázóID: A Pizzázó egyed elsődleges kulcsa.
  - Név: A pizzázó neve.
  - Elérhetőség: Összetett tulajdonság. A pizzázó elérhetőségei.

Egyedek közötti kapcsolat:

- **Pizzázó és Futár:**

  A Pizzázó és a Futár egyedek között egy a többhöz kapcsolat van, mivel egy pizzázó alkalmazhat több futárt, de egy futár csak egy pizzázónál dolgozik.

- **Pizzázó és Beszállító:**

  A Pizzázó és a Beszállító egyedek között több a többhöz kapcsolat van, mivel egy pizzázó rendelhet több beszállítótól, valamint egy beszállító beszállíthat több pizzázónak is. A kapcsolat paraméterei: a Hozzávalók, amely a beszállító által beszállított hozzávalókat jelenti, valamint a Dátum, azaz a beszállítás dátuma.

- **Pizzázó és Pizza:**

  A Pizzázó és a Pizza egyedek között egy a többhöz kapcsolat van, mivel egy pizzázónak lehet több pizzája, de egy pizza csak egy pizzázóhoz tartozhat.

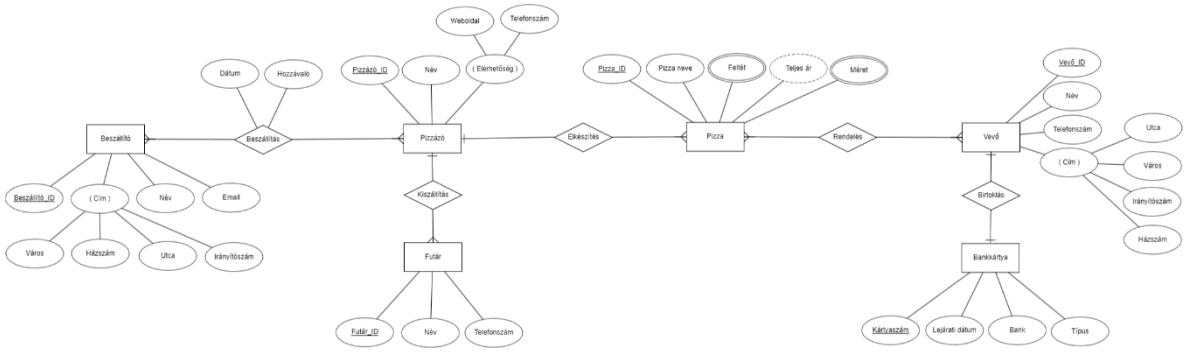
- **Pizza és Vevő:**

  A Pizza és a Vevő egyedek között több a többhöz kapcsolat van, mivel egy vevő rendelhet többfajta pizzát, és a pizzából rendelhet több különböző vevő is.

- **Vevő és Bankkártya:**

  A Vevő és a Bankkártya egyedek között egy-egy kapcsolat van, mivel egy vevőnek csak egy bankkártyája lehet, és egy bankkártyának nem lehet több tulajdonosa.

A feladat ER modellje:



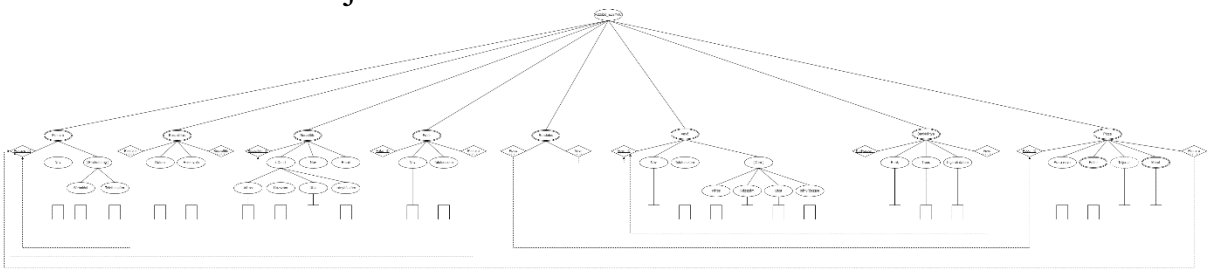
Az egyedek közötti kapcsolat

- Beszállítás kapcsolat: Több pizzázóhoz több beszállítás tartozik N:N
- Kiszállítás kapcsolat: Egy pizzázóhoz több futár tartozik: 1:N
- Elkészítés kapcsolat: Egy pizzázóhoz több pizza tartozik: 1:N
- Rendelés kapcsolat: Több pizzához több vevő tartozik: N:N
- Birtoklás kapcsolat: Egy vevőhöz egy bankkártya tartozik: 1:1

1b) Az ER-modell konvertálása XDM modellre

XDM modellnél háromféle jelölést alkalmazhatunk. Ezek az ellipszis, a rombusz, illetve a téglalap. Az ellipszis jelöli az elemeket minden egyedből elem lesz, ezen felül a tulajdonságokból is. A rombusz jelöli az attribútumokat, amelyek a kulcs tulajdonságokból keletkeznek. A téglalap jelöli a szöveget, amely majd az XML dokumentumban fog megjelenni. Azoknak az elemeknek, amelyek többször is előfordulhatnak, a jelölése dupla ellipszissel történik. Az idegenkulcsok és a kulcsok közötti kapcsolatot szaggatott vonalas nyíllal jelöljük.

A feladat XDM modellje:



1c) XML dokumentum készítése

Az XDM modell alapján az XML dokumentumot úgy készítettem el, hogy először is a root elementtel kezdtem, ami az Pizzazo_EZ3YRC volt.

A gyermek elemeiből 3-3 példányt hoztam létre, ezeknek az elemeknek az attribútumai közé tartoznak a kulcsok, illetve idegenkulcsok is, mindezek után ezeknek az elemeknek létrehoztam a többi gyermek elementet is.

XML dokumentum forráskódja

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Pizzazo_EZ3YRC xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="XMLSchemaEZ3YRC.xsd">


    <!--Pizzázók-->

    <Pizzazo pizzazo_id="1">
        <nev>Don Pepe</nev>
        <elerhetoseg>
            <weboldal>https://www.donpepe.hu/hu</weboldal>
            <telefonszam>309082091</telefonszam>
        </elerhetoseg>
    </Pizzazo>

    <Pizzazo pizzazo_id="2">
        <nev>Pizza Tábor</nev>
        <elerhetoseg>
            <weboldal>https://www.pizzatabor.hu/</weboldal>
            <telefonszam>204716655</telefonszam>
        </elerhetoseg>
    </Pizzazo>

    <Pizzazo pizzazo_id="3">
        <nev>Fortuna Pizzéria</nev>
        <elerhetoseg>
            <weboldal>https://pizzafortuna.hu/</weboldal>
```

```
            <telefonszam>302415505</telefonszam>
        </elerhetoseg>
    </Pizzazo>


    <!--Beszállítás-->

    <Beszallitas beszallito="1" pizzazo="1">
        <datum>2023-11-04</datum>
        <hozzavalo>hagyma</hozzavalo>
    </Beszallitas>

    <Beszallitas beszallito="2" pizzazo="2">
        <datum>2023-11-19</datum>
        <hozzavalo>cukor</hozzavalo>
    </Beszallitas>

    <Beszallitas beszallito="3" pizzazo="3">
        <datum>2023-11-25</datum>
        <hozzavalo>olaj</hozzavalo>
    </Beszallitas>



    <!--Beszállítók-->


    <Beszallito  beszallito_id="1">
        <nev>Kovács Lajos</nev>
        <email>mintalajos@gmail.com</email>
        <cim>
            <varos>Miskolc</varos>
            <hazszam>5</hazszam>
            <utca>Klapka György</utca>
            <iranyitoszam>3524</iranyitoszam>
        </cim>
    </Beszallito>

    <Beszallito  beszallito_id="2">
        <nev>Nehéz István</nev>
        <email>mintaistvan@gmail.com</email>
        <cim>
            <varos>Miskolc</varos>
            <hazszam>5</hazszam>
            <utca>Petőfi Sándor</utca>
            <iranyitoszam>3527</iranyitoszam>
        </cim>
    </Beszallito>
```

```xml
<Beszallito  beszallito_id="3">
    <nev>Kiss István</nev>
    <email>mintakiss@gmail.com</email>
    <cim>
        <varos>Miskolc</varos>
        <hazszam>4</hazszam>
        <utca>József Attila</utca>
        <iranyitoszam>3531</iranyitoszam>
    </cim>
</Beszallito>


<!--Futárok-->


<Futar  futar_id="1" pizzazo="1">
    <nev>Dudás Lajos</nev>
    <telefonszam>309562179</telefonszam>
</Futar>

<Futar  futar_id="2" pizzazo="2">
    <nev>Lajos Imre</nev>
    <telefonszam>309652179</telefonszam>
</Futar>

<Futar  futar_id="3" pizzazo="3">
    <nev>Nagy Márkó</nev>
    <telefonszam>209852179</telefonszam>
</Futar>


<!--Vevők-->


<Vevo vevo_id="1">
    <nev>Nehéz Gábor</nev>
    <telefonszam>306153384</telefonszam>
    <cim>
        <varos>Miskolc</varos>
        <hazszam>6</hazszam>
        <utca>Hajós Alfréd</utca>
        <iranyitoszam>3524</iranyitoszam>
    </cim>
</Vevo>


<Vevo vevo_id="2">
    <nev>Kiss Lajos</nev>
```

```
        <telefonszam>206453384</telefonszam>
        <cim>
            <varos>Miskolc</varos>
            <hazszam>4</hazszam>
            <utca>Vörösmarty Mihály</utca>
            <iranyitoszam>3532</iranyitoszam>
        </cim>
    </Vevo>

    <Vevo vevo_id="3">
        <nev>Elek János</nev>
        <telefonszam>206158484</telefonszam>
        <cim>
            <varos>Miskolc</varos>
            <hazszam>6</hazszam>
            <utca>Árok utca</utca>
            <iranyitoszam>3531</iranyitoszam>
        </cim>
    </Vevo>


    <!--Rendelés-->

    <Rendeles  pizza="1" vevo="1"></Rendeles>

    <Rendeles  pizza="2" vevo="2"></Rendeles>

    <Rendeles  pizza="3" vevo="3"></Rendeles>

    <!--Pizza-->

    <Pizza  pizza_id="1" pizzazo="1">
        <pizzaneve>Sonkás Pizza</pizzaneve>
        <feltet>paradicsom</feltet>
        <feltet>mozzarella</feltet>
        <feltet>sonka</feltet>
        <teljes_ar>2000</teljes_ar>
        <meret>17 cm</meret>
        <meret>25 cm</meret>
        <meret>30 cm</meret>
    </Pizza>

    <Pizza  pizza_id="2" pizzazo="2">
        <pizzaneve>Pizza Mexikói</pizzaneve>
        <feltet>paradicsom</feltet>
        <feltet>chili</feltet>
        <feltet>sajt</feltet>
        <teljes_ar>2550</teljes_ar>
```

```
        <meret>25 cm</meret>
        <meret>32 cm</meret>
        <meret>50 cm</meret>
    </Pizza>

    <Pizza  pizza_id="3" pizzazo="3">
        <pizzaneve>Aladdin Pizza</pizzaneve>
        <feltet>aszalt paradicsom</feltet>
        <feltet>ananász</feltet>
        <feltet>csirkemell</feltet>
        <teljes_ar>2750</teljes_ar>
        <meret>17 cm</meret>
        <meret>25 cm</meret>
        <meret>30 cm</meret>
    </Pizza>

    <!--Bankkártyák-->

    <Bankkartya  kartyaszam="1" vevo="1">
        <bank>OTP</bank>
        <tipus>bankkartya</tipus>
        <lejaratidatum>2024-05</lejaratidatum>
    </Bankkartya>

    <Bankkartya  kartyaszam="2" vevo="2">
        <bank>ERSTE</bank>
        <tipus>hitelkartya</tipus>
        <lejaratidatum>2025-06</lejaratidatum>
    </Bankkartya>

    <Bankkartya  kartyaszam="3" vevo="3">
        <bank>UNICREDIT</bank>
        <tipus>bankkartya</tipus>
        <lejaratidatum>2026-05</lejaratidatum>
    </Bankkartya>



</Pizzazo_EZ3YRC>
```

1d) XMLSchema készítése

Az XML Schemám meghatározza az adatokat, mint például a pizzázó nevét, hozzávalókat, pizza nevét stb. A telefonszamTipus azt adja meg, hogy az első karakter 1-9 közötti szám kell hogy legyen, utána a következő 8 karakter 0-9 közötti számok lehetnek. A bankTipus csak a felsorolt bankok közül enged választani (pl. OTP, KH).A ttipus pedig csak hitelkártya illetve bankkártya közül enged választani. Továbbá komplex típusokat is definiál, mint a pizzazoTipus, beszallitasTipus, beszallitoTipus, futarTipus, vevoTipus, rendelesTipus, pizzaTipus, bankkartyaTipus melyek különféle attribútumokat és elemeket tartalmaznak. Az adatbázis integritásának megőrzése érdekében elsődleges (PK) és idegen kulcsok (FK) meghatározására kerül sor, valamint egyediség biztosítása (pl. minden vevőnek egyedülálló bankkártyája lehet) az vevo_bankkartya_egyegy elem esetében. Az XML séma így biztosítja, hogy az adatok szerkezete és kapcsolatai érvényesek és következetesek legyenek.
Az XMLSchema forráskódja:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <!-- Egyszeű tipusok kigyűjtése, saját tipusok meghatározása, megszorítás-
->
    <xs:element name="nev" type="xs:string"/>
    <xs:element name="datum" type="xs:date"/>
    <xs:element name="hozzavalo" type="xs:string"/>
    <xs:element name="telefonszam" type="telefonszamTipus"/>
    <xs:element name="email" type="xs:string"/>
    <xs:element name="pizzaneve" type="pizzanevTipus"/>
    <xs:element name="feltet" type="feltetTipus"/>
    <xs:element name="teljes_ar" type="xs:int"/>
    <xs:element name="meret" type="xs:string"/>
    <xs:element name="bank" type="bankTipus"/>
    <xs:element name="tipus" type="ttipus"/>
    <xs:element name="lejaratidatum" type="xs:gYearMonth"/>


    <xs:simpleType name="telefonszamTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="[1-9]{1}[0-9]{8}"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="pizzanevTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="[a-zA-Z0-9áÁéÉíÍóÓöÖőŐúÚüÜűŰ, ]+"/>
        </xs:restriction>
    </xs:simpleType>
```

```
    <xs:simpleType name="feltetTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="[a-zA-Z0-9áÁéÉíÍóÓöÖőŐúÚüÜűŰ, ]+"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="bankTipus">
        <xs:restriction base="xs:string">
            <xs:enumeration value="OTP"/>
            <xs:enumeration value="KH"/>
            <xs:enumeration value="MBH"/>
            <xs:enumeration value="ERSTE"/>
            <xs:enumeration value="UNICREDIT"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ttipus">
        <xs:restriction base="xs:string">
            <xs:enumeration value="hitelkartya"/>
            <xs:enumeration value="bankkartya"/>
        </xs:restriction>
    </xs:simpleType>

    <!--Komplex tipusokhoz saját tipus meghatározása, sorrendiség, számosság
etc. -->

    <xs:complexType name="pizzazoTipus">
        <xs:sequence>
            <xs:element ref="nev"/>
            <xs:element name="elerhetoseg">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="weboldal" type="xs:string"/>
                        <xs:element name="telefonszam"
type="telefonszamTipus"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="pizzazo_id" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:complexType name="beszallitasTipus">
        <xs:sequence>
            <xs:element ref="datum"/>
            <xs:element ref="hozzavalo"/>
        </xs:sequence>
        <xs:attribute name="pizzazo" type="xs:integer" use="required"/>
```

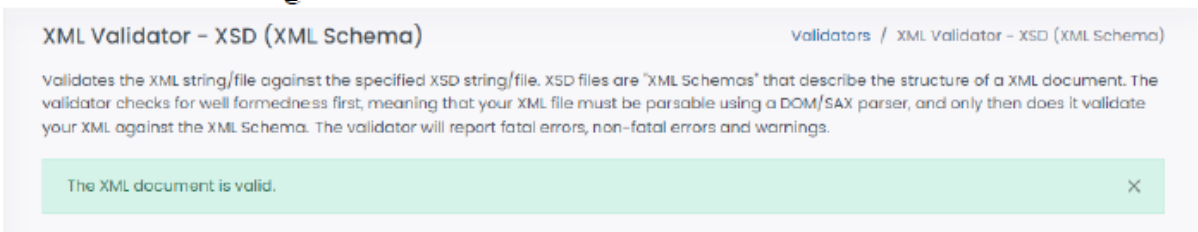
```
    <xs:attribute name="beszallito" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="beszallitoTipus">
    <xs:sequence>
        <xs:element ref="nev"/>
        <xs:element ref="email"/>
        <xs:element name="cim">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="varos" type="xs:string"/>
                    <xs:element name="hazszam" type="xs:int"/>
                    <xs:element name="utca" type="xs:string"/>
                    <xs:element name="iranyitoszam" type="xs:int"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="beszallito_id" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="futarTipus">
    <xs:sequence>
        <xs:element ref="nev"/>
        <xs:element ref="telefonszam"/>
    </xs:sequence>
    <xs:attribute name="futar_id" type="xs:integer" use="required"/>
    <xs:attribute name="pizzazo" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="vevoTipus">
    <xs:sequence>
        <xs:element ref="nev"/>
        <xs:element ref="telefonszam"/>
        <xs:element name="cim">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="varos" type="xs:string"/>
                    <xs:element name="hazszam" type="xs:int"/>
                    <xs:element name="utca" type="xs:string"/>
                    <xs:element name="iranyitoszam" type="xs:int"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="vevo_id" type="xs:integer" use="required"/>
</xs:complexType>
```

```
    <xs:complexType name="rendelesTipus">
        <xs:attribute name="pizza" type="xs:integer" use="required"/>
        <xs:attribute name="vevo" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:complexType name="pizzaTipus">
        <xs:sequence>
            <xs:element ref="pizzaneve"/>
            <xs:element ref="feltet" minOccurs="1" maxOccurs="unbounded"/>
            <xs:element name="teljes_ar" minOccurs="1" maxOccurs="unbounded"/>
            <xs:element ref="meret" minOccurs="1" maxOccurs="3"/>
        </xs:sequence>
        <xs:attribute name="pizza_id" type="xs:integer" use="required"/>
        <xs:attribute name="pizzazo" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:complexType name="bankkartyaTipus">
        <xs:sequence>
            <xs:element ref="bank"/>
            <xs:element ref="tipus"/>
            <xs:element name="lejaratidatum"/>
        </xs:sequence>
        <xs:attribute name="kartyaszam" type="xs:integer" use="required"/>
        <xs:attribute name="vevo" type="xs:integer" use="required"/>
    </xs:complexType>




    <!-- Gyökérelemtől az elemek felhasználása -->

    <xs:element name="Pizzazo_EZ3YRC">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Pizzazo" type="pizzazoTipus" minOccurs="0"
maxOccurs="100"/>
                <xs:element name="Beszallitas" type="beszallitasTipus"
minOccurs="0"
                    maxOccurs="unbounded"/>
                <xs:element name="Beszallito" type="beszallitoTipus"
minOccurs="0"
                    maxOccurs="unbounded"/>
                <xs:element name="Futar" type="futarTipus" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Vevo" type="vevoTipus" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="Rendeles" type="rendelesTipus" minOccurs="0"
maxOccurs="unbounded"/>
```

```
                <xs:element name="Pizza" type="pizzaTipus" minOccurs="0"
maxOccurs="100"/>
                <xs:element name="Bankkartya" type="bankkartyaTipus"
minOccurs="0"
                    maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>

        <!-- Elsődleges kulcsok -->

        <xs:key name="pizzazo_kulcs">
            <xs:selector xpath="Pizzazo"/>
            <xs:field xpath="@pizzazo_id"/>
        </xs:key>

        <xs:key name="beszallito_kulcs">
            <xs:selector xpath="Beszallito"/>
            <xs:field xpath="@beszallito_id"/>
        </xs:key>

        <xs:key name="futar_kulcs">
            <xs:selector xpath="Futar"/>
            <xs:field xpath="@futar_id"/>
        </xs:key>

        <xs:key name="vevo_kulcs">
            <xs:selector xpath="Vevo"/>
            <xs:field xpath="@vevo_id"/>
        </xs:key>

        <xs:key name="pizza_kulcs">
            <xs:selector xpath="Pizza"/>
            <xs:field xpath="@pizza_id"/>
        </xs:key>

        <xs:key name="bankkartya_kulcs">
            <xs:selector xpath="Bankkartya"/>
            <xs:field xpath="@kartyaszam"/>
        </xs:key>

        <!-- Idegen kulcsok -->

        <xs:keyref name="pizzazo_beszallitas_kulcs" refer="pizzazo_kulcs">
            <xs:selector xpath="Beszallitas"/>
            <xs:field xpath="@pizzazo"/>
        </xs:keyref>
```

```
        <xs:keyref name="beszallito_beszallitas_kulcs"
refer="beszallito_kulcs">
            <xs:selector xpath="Beszallitas"/>
            <xs:field xpath="@beszallito"/>
        </xs:keyref>

        <xs:keyref name="pizzazo_futar_kulcs" refer="pizzazo_kulcs">
            <xs:selector xpath="Futar"/>
            <xs:field xpath="@pizzazo"/>
        </xs:keyref>

        <xs:keyref name="pizza_rendeles_kulcs" refer="pizza_kulcs">
            <xs:selector xpath="Rendeles"/>
            <xs:field xpath="@pizza"/>
        </xs:keyref>

        <xs:keyref name="vevo_rendeles_kulcs" refer="vevo_kulcs">
            <xs:selector xpath="Rendeles"/>
            <xs:field xpath="@vevo"/>
        </xs:keyref>

        <xs:keyref name="pizzazo_pizza_kulcs" refer="pizzazo_kulcs">
            <xs:selector xpath="Pizza"/>
            <xs:field xpath="@pizzazo"/>
        </xs:keyref>

        <xs:keyref name="vevo_bankkartya_kulcs" refer="vevo_kulcs">
            <xs:selector xpath="Bankkartya"/>
            <xs:field xpath="@vevo"/>
        </xs:keyref>

        <!-- Az 1:1 kapcsolat megvalósítás -->
        <xs:unique name="vevo_bankkartya_egyegy">
            <xs:selector xpath="Bankkartya"/>
            <xs:field xpath="@vevo"/>
        </xs:unique>
    </xs:element>
</xs:schema>
```

Validáció sikeressége:



2a) DOM file beolvasás

A kód egy egyszerű alkalmazást tartalmaz, amely egy XML fájlt dolgoz fel a DOM (Document Object Model) parser segítségével. Az alkalmazás a Pizzazo_EZ3YRC XML adatstruktúráját kezeli, amely pizzériával kapcsolatos információkat, rendeléseket, vevőket, beszállítókat és egyebeket tartalmaz. A kód különböző metódusokat tartalmaz, amelyek minden egyes XML elem típusát kezelik, és kiírják vagy feldolgozzák azok adatait. A kód célja a strukturált adatok könnyű és érthető módon történő beolvasása és megjelenítése a konzolon.

```
package hu.domparse.ez3yrc;

import java.io.File;
import java.io.IOException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
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 DOMReadEZ3YRC {

    public static void main(String[] args) {
        try {
            // XML fájl beolvasása és DOM objektum létrehozása
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse(new File("XMLEZ3YRC.xml"));

            document.getDocumentElement().normalize();
            System.out.println("<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n");
            System.out.println("<Pizzazo_EZ3YRC
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation=\"XMLSchemaEZ3YRC.xsd\">\n");

            // Különböző részek feldolgozása külön metódusokon keresztül
            readPizzazos(document);
            printEmptyLine();

            readBeszallitasok(document);
            printEmptyLine();

            readBeszallitok(document);
```

```
            printEmptyLine();

            readFutarok(document);
            printEmptyLine();

            readVevok(document);
            printEmptyLine();

            readRendelesek(document);
            printEmptyLine();

            readPizzak(document);
            printEmptyLine();

            readBankkartyak(document);
            printEmptyLine();

            System.out.println("\n</Pizzazo_EZ3YRC>");
        } catch (ParserConfigurationException | IOException | SAXException
e) {
            e.printStackTrace();
        }
    }

    // Pizzázók adatainak beolvasása és kiírása
    private static void readPizzazos(Document document) {
        NodeList pizzazoList = document.getElementsByTagName("Pizzazo");
        for (int temp = 0; temp < pizzazoList.getLength(); temp++) {
            Node node = pizzazoList.item(temp);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) node;
                String pizzazoId = eElement.getAttribute("pizzazo_id");
                String nev =
eElement.getElementsByTagName("nev").item(0).getTextContent();

                System.out.println("    <Pizzazo pizzazo_id=\"" + pizzazoId
+ "\">");
                printElement("nev", nev);

                NodeList elerhetosegNodeList =
eElement.getElementsByTagName("elerhetoseg");
                if (elerhetosegNodeList.getLength() > 0) {
                    Element elerhetosegElement = (Element)
elerhetosegNodeList.item(0);
                    printElement("weboldal",
elerhetosegElement.getElementsByTagName("weboldal").item(0).getTextContent(
));
                    printElement("telefonszam",
elerhetosegElement.getElementsByTagName("telefonszam").item(0).getTextConte
nt());
                }

                System.out.println("    </Pizzazo>");
            }
        }
    }

    // Beszállítások adatainak beolvasása és kiírása
    private static void readBeszallitasok(Document document) {
        NodeList beszallitasList =
```

```
document.getElementsByTagName("Beszallitas");
        for (int temp = 0; temp < beszallitasList.getLength(); temp++) {
            Node node = beszallitasList.item(temp);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) node;
                String beszallito = eElement.getAttribute("beszallito");
                String pizzazo = eElement.getAttribute("pizzazo");
                String datum =
eElement.getElementsByTagName("datum").item(0).getTextContent();
                String hozzavalo =
eElement.getElementsByTagName("hozzavalo").item(0).getTextContent();

                System.out.println("    <Beszallitas beszallito=\"" +
beszallito + "\" pizzazo=\"" + pizzazo + "\">");
                printElement("datum", datum);
                printElement("hozzavalo", hozzavalo);
                System.out.println("    </Beszallitas>");
            }
        }
    }

    // Beszállítók adatainak beolvasása és kiírása
    private static void readBeszallitok(Document document) {
        NodeList beszallitoList =
document.getElementsByTagName("Beszallito");
        for (int temp = 0; temp < beszallitoList.getLength(); temp++) {
            Node node = beszallitoList.item(temp);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) node;
                String beszallitoId =
eElement.getAttribute("beszallito_id");
                String nev =
eElement.getElementsByTagName("nev").item(0).getTextContent();
                String email =
eElement.getElementsByTagName("email").item(0).getTextContent();

                System.out.println("    <Beszallito beszallito_id=\"" +
beszallitoId + "\">");
                printElement("nev", nev);
                printElement("email", email);

                NodeList cimNodeList =
eElement.getElementsByTagName("cim");
                if (cimNodeList.getLength() > 0) {
                    Element cimElement = (Element) cimNodeList.item(0);
                    printElement("varos",
cimElement.getElementsByTagName("varos").item(0).getTextContent());
                    printElement("hazszam",
cimElement.getElementsByTagName("hazszam").item(0).getTextContent());
                    printElement("utca",
cimElement.getElementsByTagName("utca").item(0).getTextContent());
                    printElement("iranyitoszam",
cimElement.getElementsByTagName("iranyitoszam").item(0).getTextContent());
                }

                System.out.println("    </Beszallito>");
            }
        }
    }
```

```
    // Futárok adatainak beolvasása és kiírása
    private static void readFutarok(Document document) {
        NodeList futarList = document.getElementsByTagName("Futar");
        for (int temp = 0; temp < futarList.getLength(); temp++) {
            Node node = futarList.item(temp);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) node;
                String futarId = eElement.getAttribute("futar_id");
                String pizzazoId = eElement.getAttribute("pizzazo");
                String nev =
eElement.getElementsByTagName("nev").item(0).getTextContent();
                String telefonszam =
eElement.getElementsByTagName("telefonszam").item(0).getTextContent();

                System.out.println("    <Futar futar_id=\"" + futarId + "\"
pizzazo=\"" + pizzazoId + "\">");
                printElement("nev", nev);
                printElement("telefonszam", telefonszam);
                System.out.println("    </Futar>");
            }
        }
    }

    // Vevők adatainak beolvasása és kiírása
    private static void readVevok(Document document) {
        NodeList vevoList = document.getElementsByTagName("Vevo");
        for (int temp = 0; temp < vevoList.getLength(); temp++) {
            Node node = vevoList.item(temp);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) node;
                String vevoId = eElement.getAttribute("vevo_id");
                String nev =
eElement.getElementsByTagName("nev").item(0).getTextContent();
                String telefonszam =
eElement.getElementsByTagName("telefonszam").item(0).getTextContent();

                System.out.println("    <Vevo vevo_id=\"" + vevoId +
"\">");
                printElement("nev", nev);
                printElement("telefonszam", telefonszam);

                NodeList cimNodeList =
eElement.getElementsByTagName("cim");
                if (cimNodeList.getLength() > 0) {
                    Element cimElement = (Element) cimNodeList.item(0);
                    printElement("varos",
cimElement.getElementsByTagName("varos").item(0).getTextContent());
                    printElement("hazszam",
cimElement.getElementsByTagName("hazszam").item(0).getTextContent());
                    printElement("utca",
cimElement.getElementsByTagName("utca").item(0).getTextContent());
                    printElement("iranyitoszam",
cimElement.getElementsByTagName("iranyitoszam").item(0).getTextContent());
                }

                System.out.println("    </Vevo>");
            }
        }
    }
```

```
    // Rendelések adatainak beolvasása és kiírása
    private static void readRendelesek(Document document) {
        NodeList rendelesList = document.getElementsByTagName("Rendeles");
        for (int temp = 0; temp < rendelesList.getLength(); temp++) {
            Node node = rendelesList.item(temp);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) node;
                String pizzaId = eElement.getAttribute("pizza");
                String vevoId = eElement.getAttribute("vevo");

                System.out.println("    <Rendeles pizza=\"" + pizzaId + "\"
vevo=\"" + vevoId + "\"></Rendeles>");
            }
        }
    }

    // Pizzák adatainak beolvasása és kiírása
    private static void readPizzak(Document document) {
        NodeList pizzaList = document.getElementsByTagName("Pizza");
        for (int temp = 0; temp < pizzaList.getLength(); temp++) {
            Node node = pizzaList.item(temp);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) node;
                String pizzaId = eElement.getAttribute("pizza_id");
                String pizzazoId = eElement.getAttribute("pizzazo");
                String pizzaneve =
eElement.getElementsByTagName("pizzaneve").item(0).getTextContent();
                String teljes_ar =
eElement.getElementsByTagName("teljes_ar").item(0).getTextContent();

                System.out.println("    <Pizza pizza_id=\"" + pizzaId + "\"
pizzazo=\"" + pizzazoId + "\">");
                printElement("pizzaneve", pizzaneve);
                printElement("teljes_ar", teljes_ar);

                NodeList feltetNodeList =
eElement.getElementsByTagName("feltet");
                for (int i = 0; i < feltetNodeList.getLength(); i++) {
                    System.out.println("        <feltet>" +
feltetNodeList.item(i).getTextContent() + "</feltet>");
                }

                NodeList meretNodeList =
eElement.getElementsByTagName("meret");
                for (int i = 0; i < meretNodeList.getLength(); i++) {
                    System.out.println("        <meret>" +
meretNodeList.item(i).getTextContent() + "</meret>");
                }

                System.out.println("    </Pizza>");
            }
        }
    }

    // Bankkártyák adatainak beolvasása és kiírása
    private static void readBankkartyak(Document document) {
        NodeList bankkartyaList =
document.getElementsByTagName("Bankkartya");
        for (int temp = 0; temp < bankkartyaList.getLength(); temp++) {
            Node node = bankkartyaList.item(temp);
```

```
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) node;
                String kartyaszam = eElement.getAttribute("kartyaszam");
                String vevoId = eElement.getAttribute("vevo");
                String bank =
eElement.getElementsByTagName("bank").item(0).getTextContent();
                String tipus =
eElement.getElementsByTagName("tipus").item(0).getTextContent();
                String lejaratidatum =
eElement.getElementsByTagName("lejaratidatum").item(0).getTextContent();

                System.out.println("    <Bankkartya kartyaszam=\"" +
kartyaszam + "\" vevo=\"" + vevoId + "\">");
                printElement("bank", bank);
                printElement("tipus", tipus);
                printElement("lejaratidatum", lejaratidatum);
                System.out.println("    </Bankkartya>");
            }
        }
    }

    // Segédfüggvény az XML elemek kiírásához
    private static void printElement(String name, String value) {
        System.out.println("        <" + name + ">" + value + "</" + name +
">");
    }

    // Segédfüggvény egy üres sor kiírásához a konzolon
    private static void printEmptyLine() {
        System.out.println();
    }
}
```

2b) DOM adatmódosítás

A kód először beolvassa az "XMLEZ3YRC.xml" nevű XML fájlt, majd végrehajt néhány módosítást a dokumentumon, és végül kiírja az eredményt a konzolra.

1. A végrehajtott módosítások a következők:
2. A "Pizzazo" elem nevének módosítása "Fortuna Pizzéria"-ra.
3. A "Pizza" elemek méreteinek átállítása 25, 32 és 50 értékekre.
4. A második "Beszallitas" elem hozzávalójának módosítása "olaj"-ra.
5. A harmadik "Vevo" elem telefonszámának módosítása "408883091"-re.
6. A harmadik "Bankkartya" elem bankjának "OTP"-re, típusának "hitelkartya"-ra és lejárati dátumának "2028-12"-re állítása.

Végül a program kiírja az eredményt az XML dokumentumra végrehajtott összes módosítással együtt, formázva és behúzásokkal az olvashatóság javítása érdekében.

```
package hu.domparse.ez3yrc;

import javax.xml.parsers.*;
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.*;
import java.io.File;

public class DOMModifyEZ3YRC {
    // Main metódus
    public static void main(String argv[]) {
        try {
            File inputFile = new File("XMLEZ3YRC.xml");

            DocumentBuilderFactory docFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
            Document doc = docBuilder.parse(inputFile);

            // Módosítások a main függvényben
            // Módosítás 1: Pizzazo nevének módosítása
            Element firstPizzazo = (Element)
doc.getElementsByTagName("Pizzazo").item(0);

firstPizzazo.getElementsByTagName("nev").item(0).setTextContent("Fortuna
Pizzéria");

            // Módosítás 2: Pizza méreteinek átállítása
            NodeList pizzaList = doc.getElementsByTagName("Pizza");
```

```
            for (int i = 0; i < pizzaList.getLength(); i++) {
                Element pizzaElement = (Element) pizzaList.item(i);
                NodeList meretList =
pizzaElement.getElementsByTagName("meret");
                meretList.item(0).setTextContent("25");
                meretList.item(1).setTextContent("32");
                meretList.item(2).setTextContent("50");
            }

            // Módosítás 3: Beszállítás hozzávaló módosítása
            Element beszallitasElement = (Element)
doc.getElementsByTagName("Beszallitas").item(1);

beszallitasElement.getElementsByTagName("hozzavalo").item(0).setTextContent
("olaj");

            // Módosítás 4: Vevő telefonszámának módosítása
            Element thirdVevo = (Element)
doc.getElementsByTagName("Vevo").item(2);

thirdVevo.getElementsByTagName("telefonszam").item(0).setTextContent("40888
3091");

            // Módosítás 5: Bankkártya adatainak módosítása
            Element thirdBankkartya = (Element)
doc.getElementsByTagName("Bankkartya").item(2);

thirdBankkartya.getElementsByTagName("bank").item(0).setTextContent("OTP");

thirdBankkartya.getElementsByTagName("tipus").item(0).setTextContent("hitel
kartya");

thirdBankkartya.getElementsByTagName("lejaratidatum").item(0).setTextConten
t("2028-12");

            // Kimenet kiírása
            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            DOMSource source = new DOMSource(doc);
            StreamResult consoleResult = new StreamResult(System.out);
            transformer.transform(source, consoleResult);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

2c) DOM adat lekérdezés

A különböző lekérdezéseket külön metódusok implementálják, és ezeket hívja meg a main metódus.

Az egyes lekérdezések a következők:

1. query2024UtanLejaroBankkartyak: Kilistázza azokat a bankkártyákat, amelyeknek a lejárati dátuma 2024 után van.
2. query3000NelOlcsobbPizzak: Kilistázza azokat a pizzákat, amelyek teljes ára 2600-nál olcsóbb.
3. queryMiskolciPizzeriak: Kilistázza a miskolci pizzériák nevét és elérhetőségeit.
4. queryFortunaPizzeriaHozzavaloi: Kilistázza a Fortuna Pizzéria által rendelt hozzávalókat és a beszállítókat.
5. queryDonPepeRendeltPizzak: Kilistázza a Don Pepe pizzériában rendelt pizzákat és azok teljes árát.

Minden lekérdezésnél a megfelelő XML elemeket keresi meg a DOM segítségével, majd a kívánt adatokat kiírja a konzolra.

A main metódusban az összes lekérdezést egymás után hívja meg a dokumentumon, és az esetleges kivételeket (pl. XML fájl olvasása során fellépő hiba) elkapja, majd kiírja a hibaüzeneteket a konzolra (e.printStackTrace()).

```
package hu.domparse.ez3yrc;

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

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DOMQueryEZ3YRC {

    public static void main(String[] args) {
        try {
            DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document document = db.parse("XMLEZ3YRC.xml");
```

```
            // Példa: 2024 utáni lejárattal rendelkező bankkártyák kiírása
            query2024UtanLejaroBankkartyak(document);

            // Példa: 2024 utáni lejárattal rendelkező bankkártyák kiírása
            query3000NelOlcsobbPizzak(document);

            // Példa: Kilistázza a miskolci pizzériákat
            queryMiskolciPizzeriak(document);

            // Példa: Kilistázza a Fortuna pizzéria hozzávalóit és a
beszállítót
            queryFortunaPizzeriaHozzavaloi(document);

            // Példa: Kilistázza a Don Pepe pizzéria pizzáit és azok teljes
árait
            queryDonPepeRendeltPizzak(document);

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

    // 2024 utáni lejáratú bankkártyák kiírása
    private static void query2024UtanLejaroBankkartyak(Document document) {
        System.out.println("\n");
        System.out.println("=== 2024 utáni lejárattal rendelkező
bankkártyák ===");

        NodeList bankkartyaList =
document.getElementsByTagName("Bankkartya");

        for (int i = 0; i < bankkartyaList.getLength(); i++) {
            Node bankkartyaNode = bankkartyaList.item(i);

            if (bankkartyaNode.getNodeType() == Node.ELEMENT_NODE) {
                Element bankkartyaElement = (Element) bankkartyaNode;

                String lejaratDatumString =
bankkartyaElement.getElementsByTagName("lejaratidatum").item(0).getTextCont
ent();

                try {
                    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM");
                    Date lejaratDatum = sdf.parse(lejaratDatumString);

                    // Ellenőrizzük, hogy a lejárati dátum 2024 után jár-e
le
                    if (lejaratDatum.after(sdf.parse("2025-01"))) {
                        // Ha igen, akkor kiírjuk az adatokat
                        String kartyaszam =
bankkartyaElement.getAttribute("kartyaszam");
                        String bank =
bankkartyaElement.getElementsByTagName("bank").item(0).getTextContent();
                        String tipus =
bankkartyaElement.getElementsByTagName("tipus").item(0).getTextContent();

                        System.out.println("  <Bankkartya>");
                        System.out.println("    <Kartyaszam>" + kartyaszam
+ "</Kartyaszam>");
```

```
                        System.out.println("    <Bank>" + bank +
"</Bank>");
                        System.out.println("    <Tipus>" + tipus +
"</Tipus>");
                        System.out.println("    <Lejarat>" +
lejaratDatumString + "</Lejarat>");
                        System.out.println("  </Bankkartya>");
                    }
                } catch (ParseException e) {
                    e.printStackTrace();
                }
            }
        }

        System.out.println("\n");
    }

    // 3000-nél olcsóbb pizzák kiírása
    private static void query3000NelOlcsobbPizzak(Document document) {
        System.out.println("=== 2600-nál olcsóbb pizzák ===");

        NodeList pizzaList = document.getElementsByTagName("Pizza");

        for (int i = 0; i < pizzaList.getLength(); i++) {
            Node pizzaNode = pizzaList.item(i);

            if (pizzaNode.getNodeType() == Node.ELEMENT_NODE) {
                Element pizzaElement = (Element) pizzaNode;

                // Olvassuk be a teljes árat és ellenőrizzük, hogy 2600-nál
olcsóbb
                int teljesAr =
Integer.parseInt(pizzaElement.getElementsByTagName("teljes_ar").item(0).get
TextContent());

                if (teljesAr < 2600) {
                    // Ha a pizza olcsóbb, akkor kiírjuk az adatokat
                    String pizzaId = pizzaElement.getAttribute("pizza_id");
                    String pizzazoId =
pizzaElement.getAttribute("pizzazo");
                    String pizzaneve =
pizzaElement.getElementsByTagName("pizzaneve").item(0).getTextContent();

                    System.out.println("  <Pizza>");
                    System.out.println("    <PizzaID>" + pizzaId +
"</PizzaID>");
                    System.out.println("    <PizzazoID>" + pizzazoId +
"</PizzazoID>");
                    System.out.println("    <Pizzaneve>" + pizzaneve +
"</Pizzaneve>");
                    System.out.println("    <TeljesAr>" + teljesAr +
"</TeljesAr>");
                    System.out.println("  </Pizza>");
                }
            }
        }

        System.out.println("\n");
    }
```

```
    private static void queryMiskolciPizzeriak(Document document) {
        System.out.println("=== Miskolci pizzériák neve és elérhetősége
===");

        NodeList pizzazoList = document.getElementsByTagName("Pizzazo");

        for (int i = 0; i < pizzazoList.getLength(); i++) {
            Node pizzazoNode = pizzazoList.item(i);

            if (pizzazoNode.getNodeType() == Node.ELEMENT_NODE) {
                Element pizzazoElement = (Element) pizzazoNode;

                String pizzazoId =
pizzazoElement.getAttribute("pizzazo_id");
                String nev =
pizzazoElement.getElementsByTagName("nev").item(0).getTextContent();
                String weboldal =
pizzazoElement.getElementsByTagName("weboldal").item(0).getTextContent();
                String telefonszam =
pizzazoElement.getElementsByTagName("telefonszam").item(0).getTextContent()
;

                System.out.println("  <Pizzazo>");
                System.out.println("    <PizzazoID>" + pizzazoId +
"</PizzazoID>");
                System.out.println("    <Nev>" + nev + "</Nev>");
                System.out.println("    <Weboldal>" + weboldal +
"</Weboldal>");
                System.out.println("    <Telefonszam>" + telefonszam +
"</Telefonszam>");
                System.out.println("  </Pizzazo>");
            }
        }

        System.out.println("\n");
    }

    private static void queryFortunaPizzeriaHozzavaloi(Document document) {
        System.out.println("=== Fortuna Pizzéria által rendelt hozzávalók
és beszállítóik ===");

        NodeList beszallitasList =
document.getElementsByTagName("Beszallitas");

        for (int i = 0; i < beszallitasList.getLength(); i++) {
            Node beszallitasNode = beszallitasList.item(i);

            if (beszallitasNode.getNodeType() == Node.ELEMENT_NODE) {
                Element beszallitasElement = (Element) beszallitasNode;

                String pizzazoId =
beszallitasElement.getAttribute("pizzazo");
                if (pizzazoId.equals("3")) {
                    String datum =
beszallitasElement.getElementsByTagName("datum").item(0).getTextContent();
                    String hozzavalo =
beszallitasElement.getElementsByTagName("hozzavalo").item(0).getTextContent
();
                    String beszallitoId =
beszallitasElement.getAttribute("beszallito");
```

```
                    // Beszállító adatainak lekérdezése
                    NodeList beszallitoList =
document.getElementsByTagName("Beszallito");
                    for (int j = 0; j < beszallitoList.getLength(); j++) {
                        Node beszallitoNode = beszallitoList.item(j);

                        if (beszallitoNode.getNodeType() ==
Node.ELEMENT_NODE) {
                            Element beszallitoElement = (Element)
beszallitoNode;

                            if
(beszallitoElement.getAttribute("beszallito_id").equals(beszallitoId)) {
                                String beszallitoNev =
beszallitoElement.getElementsByTagName("nev").item(0).getTextContent();

                                System.out.println("  <Beszallitas>");
                                System.out.println("    <Datum>" + datum +
"</Datum>");
                                System.out.println("    <Hozzavalo>" +
hozzavalo + "</Hozzavalo>");
                                System.out.println("    <BeszallitoID>" +
beszallitoId + "</BeszallitoID>");
                                System.out.println("    <BeszallitoNev>" +
beszallitoNev + "</BeszallitoNev>");
                                System.out.println("  </Beszallitas>");
                            }
                        }
                    }
                }
            }
        }

        System.out.println("\n");
    }

    private static void queryDonPepeRendeltPizzak(Document document) {
        System.out.println("=== Don Pepe által rendelt pizzák neve és
teljes ára ===");

        NodeList rendelesList = document.getElementsByTagName("Rendeles");

        for (int i = 0; i < rendelesList.getLength(); i++) {
            Node rendelesNode = rendelesList.item(i);

            if (rendelesNode.getNodeType() == Node.ELEMENT_NODE) {
                Element rendelesElement = (Element) rendelesNode;

                String pizzazoId = rendelesElement.getAttribute("pizza");
                if (pizzazoId.equals("1")) {
                    // Pizza adatainak lekérdezése
                    NodeList pizzaList =
document.getElementsByTagName("Pizza");
                    for (int j = 0; j < pizzaList.getLength(); j++) {
                        Node pizzaNode = pizzaList.item(j);

                        if (pizzaNode.getNodeType() == Node.ELEMENT_NODE) {
                            Element pizzaElement = (Element) pizzaNode;
```

```
                            if
(pizzaElement.getAttribute("pizza_id").equals(pizzazoId)) {
                                String pizzanev =
pizzaElement.getElementsByTagName("pizzaneve").item(0).getTextContent();
                                String teljesAr =
pizzaElement.getElementsByTagName("teljes_ar").item(0).getTextContent();

                                System.out.println("  <Rendeles>");
                                System.out.println("    <Pizzanev>" +
pizzanev + "</Pizzanev>");
                                System.out.println("    <TeljesAr>" +
teljesAr + "</TeljesAr>");
                                System.out.println("  </Rendeles>");
                            }
                        }
                    }
                }
            }
        }

    }
}
```

2d) DOM adatírás

A program beolvassa az "XMLEZ3YRC.xml" nevű XML fájlt, majd kiírja annak tartalmát egy új fájlba ("XMLEZ3YRC1.xml"). A kimeneti fájlban az XML elemek hierarchikus szerkezetben jelennek meg, és az attribútumok is megjelennek.

A printNode metódus felelős az XML dokumentum feldolgozásáért és kiírásáért. Az XML elemek hierarchikus szerkezetét és azok attribútumait kezeli, valamint az XML szöveges tartalmát is kiírja a megfelelő indentálással. A writeDocumentToFile metódus segítségével pedig az előfeldolgozott XML dokumentumot kiírja a megadott kimeneti fájlba.

A program végén kiírja a konzolra a sikerességi üzenetet: "The content has been written to the output file successfully." Ha valamilyen hiba történik (pl. XML fájl olvasása, írása, vagy transzformáció közben), akkor a kivételeket elkapja, és kiírja a hibaüzeneteket a konzolra (e.printStackTrace()).

```
package hu.domparse.ez3yrc;

import org.w3c.dom.*;
import org.xml.sax.SAXException;

import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.*;

public class DOMWriteEZ3YRC {

    public static void main(String[] args) {
        try {
            // Beolvassuk az XML fájlt
            File inputFile = new File("XMLEZ3YRC.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            // Kiírjuk a dokumentumot a konzolra
            printNode(doc.getDocumentElement(), "");

            // Kiírjuk a dokumentumot egy új XML fájlba
            writeDocumentToFile(doc, "XMLEZ3YRC1.xml");

            System.out.println("The content has been written to the output
file successfully.");
        } catch (SAXException | IOException | ParserConfigurationException
| TransformerException e) {
            // Kivétel esetén kiírjuk a hibaüzenetet
            e.printStackTrace();
        }
```

```
    }

    // Rekurzív metódus az XML fa kiírására
    private static void printNode(Node node, String indent) {
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            // Kiírjuk az XML elem nevét
            System.out.print("\n" + indent + "<" + node.getNodeName());
            if (node.hasAttributes()) {
                // Kiírjuk az XML elem attribútumait
                NamedNodeMap nodeMap = node.getAttributes();
                for (int i = 0; i < nodeMap.getLength(); i++) {
                    Node attr = nodeMap.item(i);
                    System.out.print(" " + attr.getNodeName() + "=\"" +
attr.getNodeValue() + "\"");
                }
            }

            NodeList children = node.getChildNodes();
            if (children.getLength() == 1 && children.item(0).getNodeType()
== Node.TEXT_NODE) {
                // Ha csak egy szöveges tartalom van az XML elemen belül,
kiírjuk azt
                System.out.print(">" +
children.item(0).getTextContent().trim());
                System.out.println("</" + node.getNodeName() + ">");
            } else {
                // Ha az XML elemen belül további gyerekelemek vannak,
rekurzívan kiírjuk azokat
                System.out.println(">");
                for (int i = 0; i < children.getLength(); i++)
                    printNode(children.item(i), indent + "    ");
                System.out.println(indent + "</" + node.getNodeName() +
">");
            }
        } else if (node.getNodeType() == Node.TEXT_NODE) {
            // Ha a node egy szöveges tartalom, kiírjuk azt
            String content = node.getTextContent().trim();
            if (!content.isEmpty())
                System.out.print(content);
        }
    }

    // Metódus a dokumentum kiírására egy XML fájlba
    private static void writeDocumentToFile(Document doc, String filename)
throws TransformerException {
        TransformerFactory transformerFactory =
TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        // Beállítjuk a kimeneti fájlt és formázást
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new File(filename));
        // Transformáljuk a DOM forrását a kimeneti fájlba
        transformer.transform(source, result);
    }
}
```