JEGYZŐKÖNYV

Webes adatkezelő környezetek

Féléves feladat

Gyárak

Készítette: Szabó Marcell

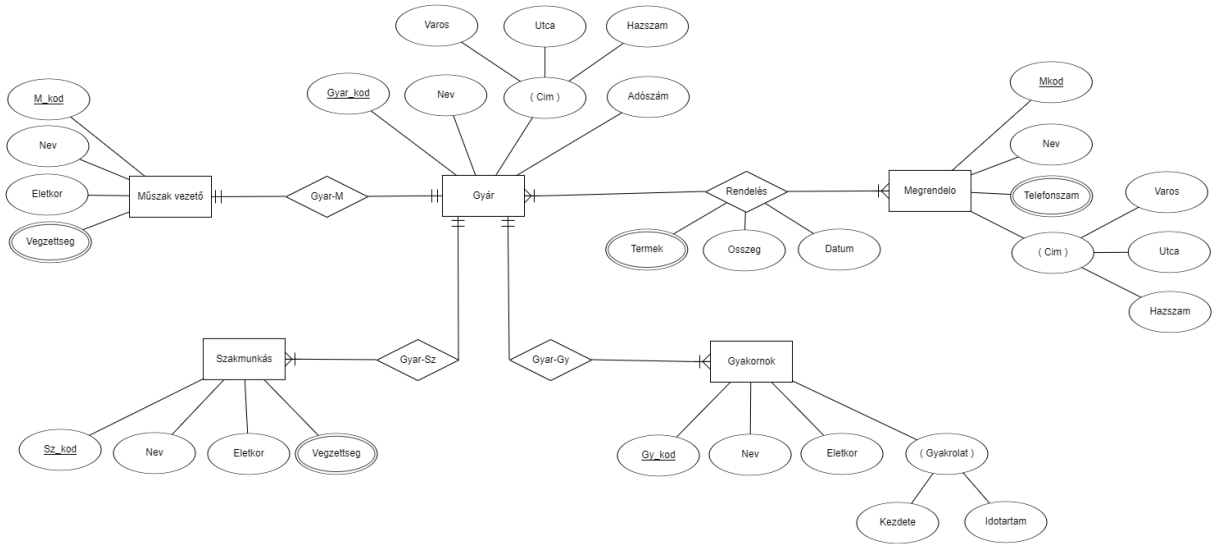
Neptunkód: FU7OMC

Dátum: 2024.11.08.

Tartalomjegyzék

1. Feladat leírása

1.1 Az adatbázis ER modell tervezése



Az egyedek tulajdonságai:

- **Gyár és Műszakvezető:**
  A Gyár és a Műszakvezető között 1:N típusú kapcsolat van, mivel egy gyár több műszakvezetőt is alkalmazhat, de egy műszakvezető csak egy gyárban dolgozhat. Például a SínCsináló Kft.-nél Trombitás Elemér és Nagy Ferenc is műszakvezetőként dolgozik.

- **Gyár és Szakmunkás:**
  A Gyár és a Szakmunkás között 1:N típusú kapcsolat van, mivel egy gyárban több szakmunkás is dolgozhat, de egy szakmunkás csak egy adott gyár alkalmazottja lehet. Például Szabó Gábor és Kovács László (Villanyszerelő) is a SínCsináló Kft.-nél dolgozik.

- **Gyár és Gyakornok:**
  A Gyár és a Gyakornok között 1:N típusú kapcsolat van, mivel egy gyár több gyakornokot is fogadhat, de egy gyakornok csak egy gyárban végezheti gyakorlatát. Illés Dániel és Hegedűs Emese például a SínCsináló Kft.-nél tölti gyakorlati idejét.

- **Gyár és Megrendelő:**
  A Gyár és a Megrendelő között 1:N típusú kapcsolat van, mivel egy gyár több megrendelővel is kapcsolatban állhat, de egy megrendelő egy adott gyártól rendel. Például a SínCsináló Kft. megrendelői között van a Máv Zrt. és a Volánbusz Zrt.

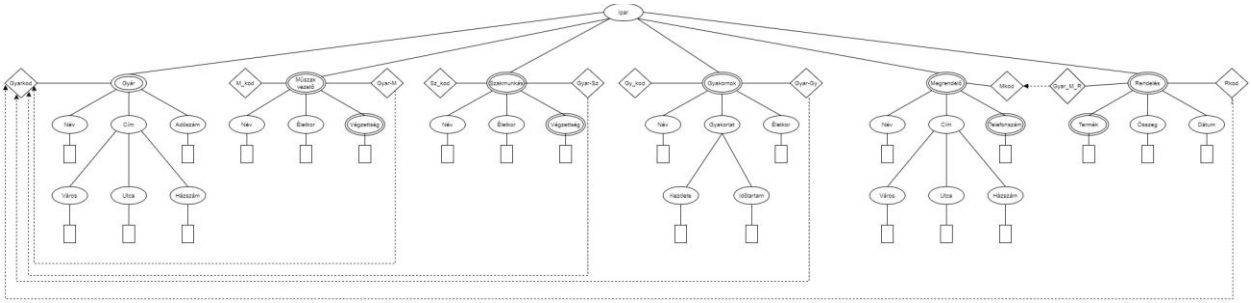
- **Rendelés és Termék:**
  A Rendelés és a Termék között 1:N típusú kapcsolat van, mivel egy rendelésben több termék is szerepelhet, de egy termék csak egy adott rendeléshez tartozik. Például a Rendelés R1-ben Vasúti Sín és Csavar is szerepel.

- **Gyár és Rendelés:**
  A Gyár és a Rendelés között 1:N típusú kapcsolat van, mivel egy gyár több rendelést is teljesíthet, de egy rendelés csak egy gyártól származik. A SínCsináló Kft. például több rendelést is kapott, mint R1 és R2.

- **Rendelés és Megrendelő:**
  A Rendelés és a Megrendelő között N:1 típusú kapcsolat van, mivel egy rendelést egy megrendelő adhat le, de egy megrendelő több rendelést is leadhat. Például a Máv Zrt. is és a Volánbusz Zrt. is adott már le különböző rendeléseket.

1.2 Az adatbázis konvertálása XDM modellre

XDM modellnél háromféle jelölés használunk: ellipszist, rombuszt és téglalapot. Az ellipszis jelöli az elemeket, minden egyedből elem lesz, illetve 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.



1.3 Az XDM modell alapján XML dokumentum készítése

Az XDM modell alapján készítettem el az XML dokumentumot. Legelőször a gyökérelemmel kezdtem, amelynek a **„XMLFU7OMC.xml”** nevet adtam. Ezek után a gyermekelemeiből eltérő módon hoztam létre példányokat.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Ipar xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="XMLSchemaFU7OMC.xsd">
    <Gyar Gyarkod="GYR1">
```

```
        <Nev>SínCsináló Kft.</Nev>
        <Cim>
            <Varos>Miskolc</Varos>
            <Utca>Szép lány</Utca>
            <Hazszam>54</Hazszam>
        </Cim>
        <Adoszam>111111111-1-11</Adoszam>
        <!-- Több műszakvezető -->
        <MuszakVezeto M_kod="M1">
            <Nev>Trombitás Elemér</Nev>
            <Vegzettseg>Lakatos</Vegzettseg>
            <Vegzettseg>Hegesztő</Vegzettseg>
            <Eletkor>62</Eletkor>
        </MuszakVezeto>
        <MuszakVezeto M_kod="M2">
            <Nev>Nagy Ferenc</Nev>
            <Vegzettseg>Gépész</Vegzettseg>
            <Eletkor>55</Eletkor>
        </MuszakVezeto>

        <!-- Több szakmunkás -->
        <Szakmunkas Sz_kod="SZ1">
            <Nev>Szabó Gábor</Nev>
            <Vegzettseg>Kisegítő</Vegzettseg>
            <Vegzettseg>Hegesztő</Vegzettseg>
            <Vegzettseg>Virágkötő</Vegzettseg>
            <Eletkor>30</Eletkor>
        </Szakmunkas>
        <Szakmunkas Sz_kod="SZ2">
            <Nev>Kovács László</Nev>
            <Vegzettseg>Villanyszerelő</Vegzettseg>
            <Vegzettseg>Kisegítő</Vegzettseg>
            <Eletkor>40</Eletkor>
        </Szakmunkas>
        <Szakmunkas Sz_kod="SZ3">
            <Nev>Kovács László</Nev>
            <Vegzettseg>Hegesztő</Vegzettseg>
            <Eletkor>50</Eletkor>
        </Szakmunkas>

        <!-- Több gyakornok -->
        <Gyakornok Gy_kod="Gyak1">
            <Nev>Illés Dániel</Nev>
```

```
            <Eletkor>20</Eletkor>
            <Gyakorlat>
                <Kezdete>2024.02.14</Kezdete>
                <Idotartam>8 hét</Idotartam>
            </Gyakorlat>
        </Gyakornok>
        <Gyakornok Gy_kod="Gyak2">
            <Nev>Hegedűs Emese</Nev>
            <Eletkor>22</Eletkor>
            <Gyakorlat>
                <Kezdete>2024.02.14</Kezdete>
                <Idotartam>8 hét</Idotartam>
            </Gyakorlat>
        </Gyakornok>
        <!-- Több megrendelő -->
        <Megrendelo Mkod="M1">
            <Nev>Máv Zrt.</Nev>
            <Cim>
                <Varos>Budapest</Varos>
                <Utca>Máv</Utca>
                <Hazszam>32</Hazszam>
            </Cim>
            <Telefonszam>+36 20 456 4353</Telefonszam>
        </Megrendelo>
        <Megrendelo Mkod="M2">
            <Nev>Volánbusz Zrt.</Nev>
            <Cim>
                <Varos>Debrecen</Varos>
                <Utca>Nagyerdei körút</Utca>
                <Hazszam>5</Hazszam>
            </Cim>
            <Telefonszam>+36 30 123 4567</Telefonszam>
        </Megrendelo>
        <!-- Több termék a rendelésben -->
        <Rendeles GyarMR="GYR1" Rkod="R1">
            <Termek>Vasúti Sín</Termek>
            <Termek>Acél Lemez</Termek>
            <Termek>Csavar</Termek>
            <Osszeg>12345654.00</Osszeg>
            <Datum>2024-03-21</Datum>
        </Rendeles>
        <Rendeles GyarMR="GYR1" Rkod="R2">
            <Termek>Betongerenda</Termek>
            <Termek>Acélcső</Termek>
```

```
            <Osszeg>5789000.00</Osszeg>
            <Datum>2024-04-10</Datum>
        </Rendeles>
    </Gyar>
</Ipar>
```

1.4 Az XML dokumentum alapján XMLSchema készítése

Az XML dokumentum validálásának megkönnyítésére egy sémát készítettem. Először definiáltam az egyszerű típusokat, majd létrehoztam saját típusokat, összesen ötöt. Például a telefonszámhoz egy olyan típust készítettem, amely reguláris kifejezéssel ellenőrzi a formátumot, és egy másik típust az érvényes kódok listájával enumeráció segítségével. Ezután megterveztem a komplex típusokat az összes elemre, beleértve az összetett cím és személyzet adatstruktúrákat. Végül definiáltam elsődleges kulcsokat az egyedi azonosításhoz, valamint idegen kulcsokat az elemek közötti kapcsolatok biztosítására. A rendszerben megvalósítottam egy 1:N kapcsolatot a Gyár és a Rendelés között, ahol egy gyárhoz több rendelés is tartozhat.

XMLSchemaFU7OMC.xsd:

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

    <!-- Egyszerű típus a korhoz -->
    <xs:simpleType name="korTipus">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1" />
            <xs:maxExclusive value="100" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Cím komplex típus -->
    <xs:complexType name="CimTipus">
        <xs:sequence>
            <xs:element name="Varos" type="xs:string" />
            <xs:element name="Utca" type="xs:string" />
            <xs:element name="Hazszam" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <!-- Alapvető személy típus -->
    <xs:complexType name="SzemelyTipus">
        <xs:sequence>
            <xs:element name="Nev" type="xs:string" />
```

```
            <xs:element name="Vegzettseg" type="xs:string" minOccurs="1"
maxOccurs="unbounded"/>
            <xs:element name="Eletkor" type="korTipus" />
        </xs:sequence>
    </xs:complexType>

    <!-- Műszakvezető típus -->
    <xs:complexType name="MuszakVezetoTipus">
        <xs:complexContent>
            <xs:extension base="SzemelyTipus">
                <xs:attribute name="M_kod" type="xs:string" use="required" />
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Szakmunkás típus -->
    <xs:complexType name="SzakmunkasTipus">
        <xs:complexContent>
            <xs:extension base="SzemelyTipus">
                <xs:attribute name="Sz_kod" type="xs:string" use="required" />
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>

    <!-- Gyakornok típus -->
    <xs:complexType name="GyakornokTipus">
        <xs:sequence>
            <xs:element name="Nev" type="xs:string" />
            <xs:element name="Eletkor" type="korTipus" />
            <xs:element name="Gyakorlat">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Kezdete" type="xs:string" />
                        <xs:element name="Idotartam" type="xs:string" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="Gy_kod" type="xs:string" use="required" />
    </xs:complexType>

    <!-- Megrendelő típus -->
    <xs:complexType name="MegrendeloTipus">
```

```
        <xs:sequence>
            <xs:element name="Nev" type="xs:string" />
            <xs:element name="Cim" type="CimTipus" />
            <xs:element name="Telefonszam" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="Mkod" type="xs:string" use="required" />
    </xs:complexType>

    <!-- Rendelés típus -->
    <xs:complexType name="RendelesTipus">
        <xs:sequence>
            <xs:element name="Termek" type="xs:string" minOccurs="1"
maxOccurs="unbounded" />
            <xs:element name="Osszeg" type="xs:string" />
            <xs:element name="Datum" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="Rkod" type="xs:string" use="required" />
        <xs:attribute name="GyarMR" type="xs:string" use="required" />
    </xs:complexType>

    <!-- Gyar típus -->
    <xs:complexType name="GyarTipus">
        <xs:sequence>
            <xs:element name="Nev" type="xs:string" />
            <xs:element name="Cim" type="CimTipus" />
            <xs:element name="Adoszam" type="xs:string" />
            <xs:element name="MuszakVezeto" type="MuszakVezetoTipus"
minOccurs="1"
                maxOccurs="unbounded" />
            <xs:element name="Szakmunkas" type="SzakmunkasTipus" minOccurs="1"
maxOccurs="unbounded" />
            <xs:element name="Gyakornok" type="GyakornokTipus" minOccurs="1"
maxOccurs="unbounded" />
            <xs:element name="Megrendelo" type="MegrendeloTipus" minOccurs="1"
maxOccurs="unbounded" />
            <xs:element name="Rendeles" type="RendelesTipus" minOccurs="1"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="Gyarkod" type="xs:string" use="required" />
    </xs:complexType>

    <!-- Ipar gyökérelem -->
    <xs:element name="Ipar">
```

```
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Gyar" type="GyarTipus" minOccurs="1"
maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>

        <!-- Elsődleges kulcsok -->
        <xs:key name="gyar_kulcs">
            <xs:selector xpath="Gyar" />
            <xs:field xpath="@Gyarkod" />
        </xs:key>
        <xs:key name="muszakvezeto_kulcs">
            <xs:selector xpath="Gyar/MuszakVezeto" />
            <xs:field xpath="@M_kod" />
        </xs:key>
        <xs:key name="szakmunkas_kulcs">
            <xs:selector xpath="Gyar/Szakmunkas" />
            <xs:field xpath="@Sz_kod" />
        </xs:key>

        <xs:key name="gyakornok_kulcs">
            <xs:selector xpath="Gyar/Gyakornok" />
            <xs:field xpath="@Gy_kod" />
        </xs:key>
        <xs:key name="rendeles_kulcs">
            <xs:selector xpath="Gyar/Rendeles" />
            <xs:field xpath="@Rkod" />
        </xs:key>

        <!-- Idegen kulcsok -->
        <xs:keyref name="rendeles_gyar_kulcs" refer="gyar_kulcs">
            <xs:selector xpath="Gyar/Rendeles" />
            <xs:field xpath="@GyarMR" />
        </xs:keyref>
        <xs:keyref name="muszakvezeto_gyar_kulcs" refer="gyar_kulcs">
            <xs:selector xpath="Ipar/Gyar/MuszakVezeto" />
            <xs:field xpath="@M_kod" />
        </xs:keyref>
        <xs:keyref name="szakmunkas_gyar_kulcs" refer="gyar_kulcs">
            <xs:selector xpath="Ipar/Gyar/Szakmunkas" />
            <xs:field xpath="@Sz_kod" />
        </xs:keyref>
```

```
        <xs:keyref name="gyakornok_gyar_kulcs" refer="gyar_kulcs">
            <xs:selector xpath="Ipar/Gyar/Gyakornok" />
            <xs:field xpath="@Gy_kod" />
        </xs:keyref>
        <xs:keyref name="megrendelo_gyar_kulcs" refer="gyar_kulcs">
            <xs:selector xpath="Ipar/Gyar/Megrendelo" />
            <xs:field xpath="@Mkod" />
        </xs:keyref>

    </xs:element>
</xs:schema>
```

2. DOM program készítése

2.1 Adatolvasás – fájlnév: DOMReadNeptunkod.java

main metódus:

A program belépési pontja, amely inicializálja az XML-feldolgozást:

Fájlbeolvasás: XMLFU7OMC.xml fájl betöltése.

Az XML feldolgozását külön metódusok végzik az adatok (gyárak, műszakvezetők, szakmunkások stb.) lekérdezésére és kiírására.

parseXML metódus:

Funkció: XML fájl beolvasása és DOM-parszolása.

Paraméter: File file – a beolvasandó XML fájl.

Visszatérési érték: Document – a DOM dokumentum, amely az XML adatszerkezetet reprezentálja.

queryXMLGyarkodok metódus:

Funkció: Az XML-ben található gyár adatok kiírása.

Minden Gyar elemre iterálva megjeleníti a gyár azonosítóját (Gyarkod), nevét, címét és adószámát.

queryXMLMuszakVezetok metódus:

Funkció: Műszakvezetők adatainak megjelenítése.

A MuszakVezeto elemek kiolvasása, amelyek tartalmazzák a műszakvezető kódját (M_kod), nevét, végzettségét és életkorát.

queryXMLSzakmunkasok metódus:

Funkció: Szakmunkások adatainak megjelenítése.

A Szakmunkas elemek kiolvasása, amelyek tartalmazzák a szakmunkás kódját (Sz_kod), nevét, végzettségét és életkorát.

queryXMLGyakornokok metódus:

Funkció: Gyakornokok és gyakorlatok adatainak megjelenítése.

A Gyakornok elemek kiírása, beleértve a gyakornok kódját (Gy_kod), nevét és életkorát.

A gyakornokok gyakorlati időszakainak (Gyakorlat) részletei, mint a kezdete és időtartama.

queryXMLMegrendelok metódus:

Funkció: Megrendelők adatainak kiírása.

Minden Megrendelo elem attribútumai: azonosító (Mkod), név, cím (város, utca, házszám) és telefonszám kiíratása.

queryXMLRendelesek metódus:

Funkció: Rendelések adatainak kiírása.

Rendeles elemek kiolvasása, beleértve a gyárkódot (GyarMR), rendeléskódot (Rkod), termékeket és a rendelés összegét, dátumát.

Érdekesnek talált kódrészlet:

Kódrészlet: **queryXMLGyarkodok**

```
// Gyakornokok kiírása
private static void queryXMLGyakornokok(Document document) {
        NodeList gyakornokNodeList =
document.getElementsByTagName("Gyakornok");
        for (int i = 0; i < gyakornokNodeList.getLength(); i++) {
            Node gyakornokNode = gyakornokNodeList.item(i);
            if (gyakornokNode.getNodeType() == Node.ELEMENT_NODE) {
                Element gyakornokElement = (Element) gyakornokNode;
                String Gy_kod = gyakornokElement.getAttribute("Gy_kod");
                String nev =
gyakornokElement.getElementsByTagName("Nev").item(0).getTextContent();
                String eletkor =
gyakornokElement.getElementsByTagName("Eletkor").item(0).getTextContent();

                System.out.println("-------------------------------");
                System.out.println("Gyakornok:");
                System.out.println("    GyakornokKod=\"" + Gy_kod + "\"");
                System.out.println("    Nev: " + nev);
                System.out.println("    Eletkor: " + eletkor);

                NodeList gyakorlatNodes =
gyakornokElement.getElementsByTagName("Gyakorlat");
                for (int j = 0; j < gyakorlatNodes.getLength(); j++) {
                    Element gyakorlatElement = (Element)
gyakorlatNodes.item(j);
                    String kezdet =
gyakorlatElement.getElementsByTagName("Kezdete").item(0).getTextContent();
                    String idotartam =
gyakorlatElement.getElementsByTagName("Idotartam").item(0).getTextContent();
```

```
                    System.out.println("    Gyakorlat:");
                    System.out.println("        Kezdete: " + kezdet);
                    System.out.println("        Idotartam: " + idotartam);
                }
            }
        }
    }
```

Érdekes, mert bemutatja, hogyan lehet programozott módon strukturált adatokat olvasni és feldolgozni XML-ből, amit tipikusan konfigurációk, adatintegráció során használnak.

Magyarázat:

XML Feldolgozás DOM-mal:

A **NodeList gyarNodeList = document.getElementsByTagName("Gyar");** sorral lekérjük az összes **<Gyar>** elemet az XML-ből.

Ezután iterál a **NodeList** elemein, hogy minden egyes **<Gyar>** csomópontot külön feldolgozzon.

Csomópont és elem típusok ellenőrzése:

Az **if (gyarNode.getNodeType() == Node.ELEMENT_NODE)** feltétellel ellenőrzi, hogy valóban egy **Element** típusú csomóponttal dolgozik-e, elkerülve ezzel nem kívánt csomópontok feldolgozását.

Attribútumok és alárendelt elemek kezelése:

Az **gyarElement.getAttribute("Gyarkod")** például a **<Gyar>** elem **Gyarkod** attribútumának értékét adja vissza.

A **getElementsByTagName("Nev").item(0).getTextContent()** sor pedig a **<Nev>** alárendelt elem tartalmát olvassa ki.

Adatok kiíratása:

A **System.out.println()** hívások rendezett, jól formázott kimenetet biztosítanak, így az adatok könnyen olvashatóak.

2.2 Adatírás – fájlnév: DOMWriteNeptunkod.java

writeFactoryDataToXML(): Ez a metódus létrehoz egy új XML dokumentumot, hozzáad egy gyökérelemet ("Ipar"), majd különböző elemeket (gyár, műszakvezetők, szakmunkások, gyakornokok, megrendelők, rendelések) szúr be.

Adatok hozzáadása az XML-hez:

addCimElement: Hozzáadja a gyár címét "Cim" néven, mely tartalmazza a várost, utcát és házszámot.

addMuszakVezeto: Új műszakvezető hozzáadása az adott gyárhoz.

addSzakmunkas: Szakmunkás adatainak beszúrása a gyár elem alá.

addGyakornok: Gyakornok hozzáadása, beleértve a név, életkor, gyakorlat kezdete és időtartama adatokat.

addMegrendelo: Megrendelő létrehozása és címeinek hozzáadása (város, utca, házszám, telefonszám).

addRendeles: Rendelés hozzáadása, amely termékeket, összeget és rendelési dátumot tartalmaz.

Érdekesnek talált kódrészlet:

Kódrészlet: **addRendeles**

```
// Rendelés hozzáadása
private static void addRendeles(Document document, Element gyar, String kod1,
String kod2, String termek1, String termek2, String termek3, String osszeg,
String datum) {
        Element rendeles = document.createElement("Rendeles");
        rendeles.setAttribute("GyarMR", kod1);
        rendeles.setAttribute("Rkod", kod2);

        addTextElement(document, rendeles, "Termek", termek1);
        if (!termek2.isEmpty()) addTextElement(document, rendeles, "Termek",
termek2);
        if (!termek3.isEmpty()) addTextElement(document, rendeles, "Termek",
termek3);
        addTextElement(document, rendeles, "Osszeg", osszeg);
        addTextElement(document, rendeles, "Datum", datum);

        gyar.appendChild(rendeles);
    }
```

Ez a metódus jól szemlélteti, hogyan lehet dinamikusan XML elemeket létrehozni és feltölteni attribútumokkal, valamint hogyan kezelhetjük a változó mennyiségű adatokat rugalmasan.

Magyarázat:

Elemek létrehozása és attribútumok hozzáadása:

rendeles elem jön létre, amelyhez két attribútumot (**GyarMR és Rkod)** rendelünk.

Termékek dinamikus hozzáadása:

termek1 hozzáadása kötelező, míg **termek2** és **termek3** csak akkor kerülnek hozzáadásra, ha nem üresek. Ez az ellenőrzés az **if** struktúrákkal történik.

Egyéb adatok hozzáadása:

Az **Osszeg** és **Datum** mezőket szintén hozzáadja az **addTextElement** metódus segítségével.

Elem hozzáfűzése a gyárhoz:

A **rendeles** a **gyar** elemhez kapcsolódik a **gyar.appendChild(rendeles)** sor által.

2.3 Adatlekérdezés – fájlnév: DOMQueryNeptunkod.java

DOMQueryFU7OMC osztály: Ez az osztály olyan metódusokat tartalmaz, amelyek egy XML fájl tartalmát képes beolvasni és feldolgozni DOM segítségével, ráadásul lekérdezéseket is képesek vagyunk végrehajtani.

Lekérdezések és metódusok:

QueryXMLAllMuszakVezetok:

Leírás: Az összes műszakvezető adatainak kiírása.

Művelet: Kilistázza a MuszakVezeto elemek összes adatát, beleértve a kódot, nevet, végzettséget és életkort.

QueryXMLAllSzakmunkasok:

Leírás: Az összes szakmunkás adatainak kiírása.

Művelet: Kilistázza a Szakmunkas elemek adatait, például a kódot, nevet, végzettséget és életkort.

QueryXMLAllGyakornokok:

Leírás: Az összes gyakornok adatainak kiírása.

Művelet: Kilistázza a Gyakornok elemek kódját, nevét és életkorát.

QueryXMLAllMegrendelok:

Leírás: Az összes megrendelő adatainak kiírása.

Művelet: Kiírja a Megrendelo elemek összes adatát, beleértve a kódot, nevet, várost, utcát, házszámot és telefonszámot.

Érdekesnek talált kódrészlet:

```
// Műszakvezetők kiírása
private static void queryXMLAllMuszakVezetok(Document document) {
        System.out.println("\n1. \033 Az összes műszakvezető adatainak
kiírása" );
```

```
        NodeList muszakVezetoNodeList =
document.getElementsByTagName("MuszakVezeto");
        for (int i = 0; i < muszakVezetoNodeList.getLength(); i++) {
            Node muszakVezetoNode = muszakVezetoNodeList.item(i);
            if (muszakVezetoNode.getNodeType() == Node.ELEMENT_NODE) {
                Element muszakVezetoElement = (Element) muszakVezetoNode;
                String kod = muszakVezetoElement.getAttribute("M_kod");
                String nev =
muszakVezetoElement.getElementsByTagName("Nev").item(0).getTextContent();
                String vegzettseg =
muszakVezetoElement.getElementsByTagName("Vegzettseg").item(0).getTextContent()
;
                String eletkor =
muszakVezetoElement.getElementsByTagName("Eletkor").item(0).getTextContent();

                System.out.println("<MuszakVezeto Kod=\"" + kod + "\">");
                System.out.println("    <Nev>" + nev + "</Nev>");
                System.out.println("    <Vegzettseg>" + vegzettseg +
"</Vegzettseg>");
                System.out.println("    <Eletkor>" + eletkor + "</Eletkor>");
                System.out.println("</MuszakVezeto>");
            }
        }
    }
```

Ez a rész azért tűnt ki, mert jól szemlélteti, hogyan lehet egy XML dokumentumot hierarchikusan bejárni és feldolgozni a DOM használatával.

Magyarázat:

XML Bejárása:

A metódus beolvassa az XML fájlból az összes **MuszakVezeto** elemhez tartozó adatot.

A **document.getElementsByTagName("MuszakVezeto")** utasítással lekéri az összes **MuszakVezeto** elemet egy NodeList-be.

Ezután egy ciklusban (for) bejárja az összes ilyen elemet, és ellenőrzi, hogy a csomópont tényleg **ELEMENT_NODE** típusú-e.

Elemek feldolgozása:

Az egyes **MuszakVezeto** elemek attribútumait **(M_kod)** és gyermekelemeit **(Nev, Vegzettseg, Eletkor)** külön-külön olvassa ki.

Az **getAttribute()** és az **getElementsByTagName()** metódusokkal történik az XML-adatok kinyerése.

2.4 Adatmódosítás – fájlnév: DOMModifyNeptunkod.java

DOMModifyFU7OMC osztály: Ez az osztály végzi el az XML fájlban a módosításokat a DOM használatával.

Adatmódosításhoz a következő segédmetódusokat használtam:

- **getElementsByTagName:** Node-listát ad vissza, amely az adott elem nevű összes leszármazottját tartalmazza a hívott elemen belül.
- **setTextContext:** beállítja az elem szövegtartalmát a megadott értékre.

modifyXMLElements metódus: ez a metódus fogja módosítani az 'XMLFU7OMC.xml' fájlt és megjeleníteni a konzolra.

Érdekesnek talált kódrészlet:

```
// 1. Módosítás: Műszakvezető név módosítása
NodeList muszakVezetoList = document.getElementsByTagName("MuszakVezeto");
Element muszakVezetoElement = (Element) muszakVezetoList.item(0);
muszakVezetoElement.getElementsByTagName("Nev").item(0).setTextContent("Trombit
ás Elemér");
```

Ez kódrészlet minimalista módon, de hatékonyan végzi el az XML módosítását. Egyetlen sorral egy bonyolult hierarchiában található adatot frissít.

Magyarázat:

XML elemek kiválasztása:
A **document.getElementsByTagName("MuszakVezeto")** utasítás lekéri az összes **MuszakVezeto** elemet az XML dokumentumból. Ez egy NodeList típusú objektumot hoz létre, amely egy tömbhöz hasonló struktúra, és az összes **MuszakVezeto** elemet tartalmazza.
Elem elérése:
A **muszakVezetoList.item(0)** az első **MuszakVezeto** elemet adja vissza. Az **item(0)** használata azt jelenti, hogy a lista első elemére hivatkozunk. A visszakapott Node típust Element típusra konvertáljuk, hogy további műveleteket végezhessünk rajta.
Gyermekelem módosítása:
A **muszakVezetoElement.getElementsByTagName("Nev").item(0)** a **MuszakVezeto** elem alatti első **Nev** elemet választja ki. A **setTextContent("Trombitás Elemér")** metódus pedig a **Nev** elem szöveges tartalmát módosítja, így a korábbi név helyett "Trombitás Elemér" kerül az XML-be.