JEGYZŐKÖNYV

Webes adatkezelő környezetek

Féléves feladat

Ételrendelő rendszer

Készítette: **Balla Levente Dávid**

Neptunkód: **GH5OGN**

Dátum: **2025. December**

Miskolc, 2025

Tartalom

1. A feladat leírása

Az általam kitalált féléves feladat egy ételrendelő és kiszállító rendszer adatmodelljének kidolgozása. A rendszerem célja az lesz, hogy összekösse a vásárlókat az éttermekkel és a futárokkal, lehetővé téve az ételek online megrendelését és házhoz szállítását. Az adatmodell amit kidolgoztam öt központi egyedre épül, ezek a rendszer alapvető szereplői:

-Vásárló: Ő használja a rendszert. Rendelkezik azonosító adatokkal, egyedi kulccsal (vevo_id), összetett névvel (Vezeteknev, Keresztnev), e-mail címmel, valamint többértékű tulajdonságokkal, mint a Telefonszam (ebből többet is megadhat) és a MentettCimek ahová a címeit mentheti ahová az étel rendelné(szintén többet tárolhat).

-Étterem: Az ételeket kínáló partner. Azonosító (etterem_id), név, összetett cím ami azért összetett mert a cím adatai szétbontva szerepelnek benne, valamint többértékű Kategoria (pl. Lángos, Pizza) és nyitvatartási adatokkal rendelkezik a rendszerben.

-Futár: A kiszállítást végző személy. Van azonosítója (futar_id), összetett név tulajdonsága, telefonszáma, járműazonosítója, ami a rendszáma (JarmuRendszam) és a vásárlók által adott Ertekeles tartozik hozzá, mivel a rendelés megérkezte után lehetősége lenne a vásárlóknak pontoznia a futárt.

-Étel: Egy konkrét étteremhez (etterem_idref) kapcsolt termék vagy étel. Jellemzői az azonosító (etel_id), név, ár, leírás és a többértékű Allergének, amiben az allergének felsorolhatók lennének, mivel a vásárlás szempontjából fontos információ.

-Rendelés: A rendszer központi eleme, amely összekapcsolja a többi egyedet. A rendelés (rendeles_id) egy vásárlóhoz (vevo_idref), egy étteremhez (etterem_idref) és egy futárhoz (futar_idref) van rendelve és ez mind 1:N kapcsolat. Tartalmazza a rendelés idejét, státuszát, a szállítási címet (ami eltérhet a mentett címektől) és a végösszeget.

A feladat fontos és legnehezebb eleme volt az N:M kapcsolat megléte. Ez a Rendelés és az Étel között valósul meg, mivel egy rendelés több ételt tartalmazhat, és egy étel több rendelésben is szerepelhet. Ezt az ER modellen egy tartalmaz kapcsolat, az XML-ben pedig egy beágyazott RendelesTetelei elem valósítja meg, ami Tetel elemeket tartalmaz. Ezek a tételek hivatkoznak a konkrét ételre (etel_idref), és rögzítik az adott rendeléshez tartozó darabszámot, a vásárláskori árat és az esetleges extra kéréseket, például extra szószok stb.

2. ER modell

A feladat első része az általam kitalált adatbázis tervének elkészítése volt egy ER modell formájában. A modell célja, hogy logikai szinten, részletektől függetlenül ábrázolja a rendszerben kezelt egyedeket, azok tulajdonságait és a köztük fennálló kapcsolatokat. A tervezés a draw.io szerkesztőprogrammal készült, a szabványos Chen-jelölésrendszert alkalmaztam benne.

A feladatkiírásnak megfelelően a modell minimum 5 egyedet tartalmaz. Ezek a ételrendelő rendszeremben a következők:

Vásárló: A rendszer felhasználója.

Étterem: Az ételeket kínáló partner.

Futár: A kiszállítást végző személy.

Étel: Az éttermek által kínált termék.

Rendelés: A tranzakciót képviselő központi egyed.

A modell kidolgozása során implementáltam a feladatkiírásban szereplő összes tulajdonságtípust:

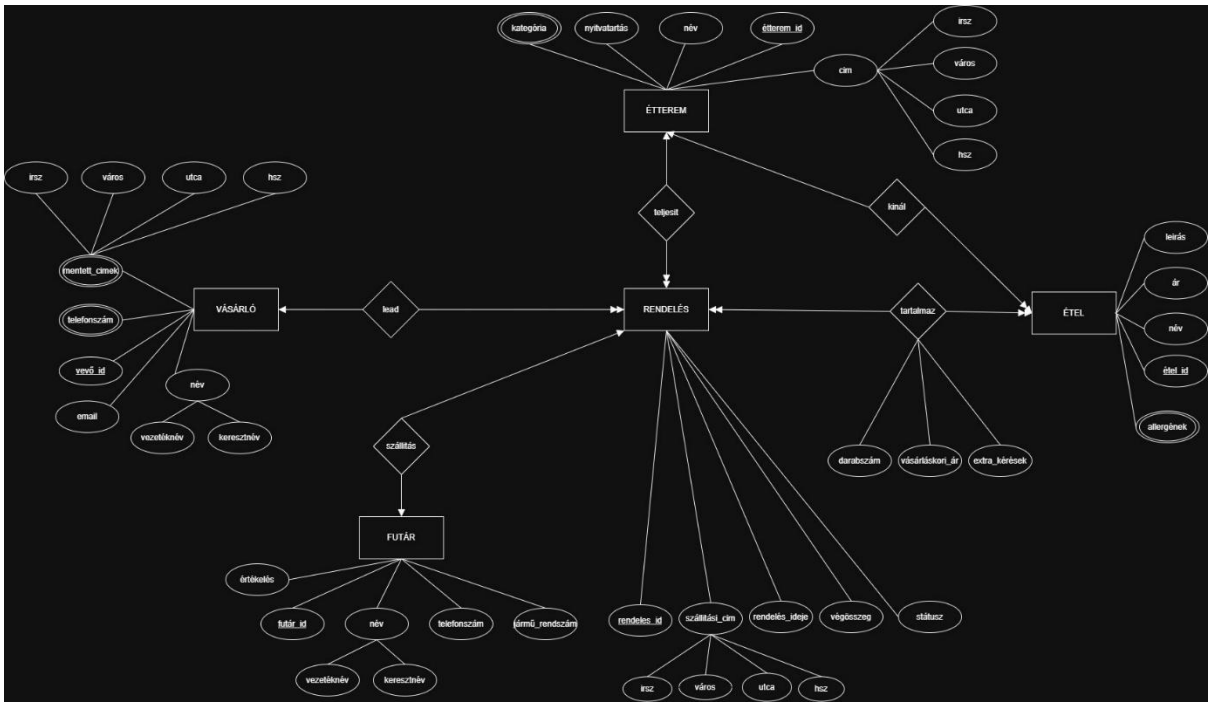
Elsődleges kulcsok (aláhúzva): Minden egyed kapott egy egyedi azonosítót (pl. vevo_id, etterem_id, rendeles_id, etel_id, futar_id).

Összetett tulajdonságok: Ahol egy tulajdonság több részből áll, ott ezt ábrázoltam. Ilyen a Név ami Vezeteknev-ből és Keresztnev-ből áll és a Cim (mely Irsz, Varos, Utca, Hsz részekre bomlik).

Többértékű tulajdonságok (dupla ellipszissel jelöltem): Ahol egy tulajdonságból több is tartozhat egy egyedhez. Ilyen a Vásárló Telefonszáma, az Étterem Kategoriája, és az Étel Allergénjei.

A kapcsolatok tekintetében a modell többféle kapcsolattípust is tartalmaz. A Vásárló-Rendelés, Étterem-Rendelés, Futár-Rendelés és Étterem-Étel mind 1:N kapcsolatok.

A feladatkiírásnak megfelelően létrehoztam egy M:N több a többhöz kapcsolatot is a Rendelés és az Étel egyedek között (tartalmaz). Ez a kapcsolat mutatja, hogy egy rendelés több ételt is tartalmazhat, és egy étel típusa több rendelésben is szerepelhet. Ennek a kapcsolategyednek a kiírás szerint saját tulajdonságai is vannak: darabszam, vasarlaskori_ar (mert az étel ára idővel változhat) és extra_keresek.



1. Ábra: az ételrendelő rendszer ER modellje(GH5OGN_ER.jpg)

3. XDM modell

Az elkészített ER modellt át kellett alakítani egy XML-specifikus, hierarchikus adatmodellé, ami az XDM. Ez a modell már nem egy relációs adatbázis, hanem egy fa-struktúrájú XML dokumentum tervrajza, ami a későbbiekben az XML megvalósításához segítségül fog szolgálni egy későbbi feladatban is. Az átalakítás célja az volt, hogy kiderüljön, melyik egyedekből lesznek elemek, melyik tulajdonságokból attribútumok vagy beágyazott elemek, és hogyan épülnek fel a kapcsolatok majd.

Elemek:

Gyökérelem: Létrehoztam egy mesterséges gyökérelemet, ami az EtelRendeloRendszer nevet kapta, ez összefogja az egész adatbázist, ez mindig szükséges.

Egyedek: A fő egyedek, mint a Vásárló, Étterem közvetlenül a gyökér alá kerültek. Belőlük mivel számtalan lehet, ezért dupla ellipszisben kaptak helyet.

Kulcsok: Az elsődleges kulcsokat, mint a vevo_id, etterem_id stb. az XDM-ben attribútumként ábrázoltam az adott elemen. Ez a legtisztább megoldás az azonosítók tárolására.

Tulajdonságok:

Az egyszerű tulajdonságok, mint az email és az ár egyszerű beágyazott gyermekelemek lettek.

Az összetett tulajdonságok pl. Név, Cim beágyazott, komplex gyermekelemek lettek.

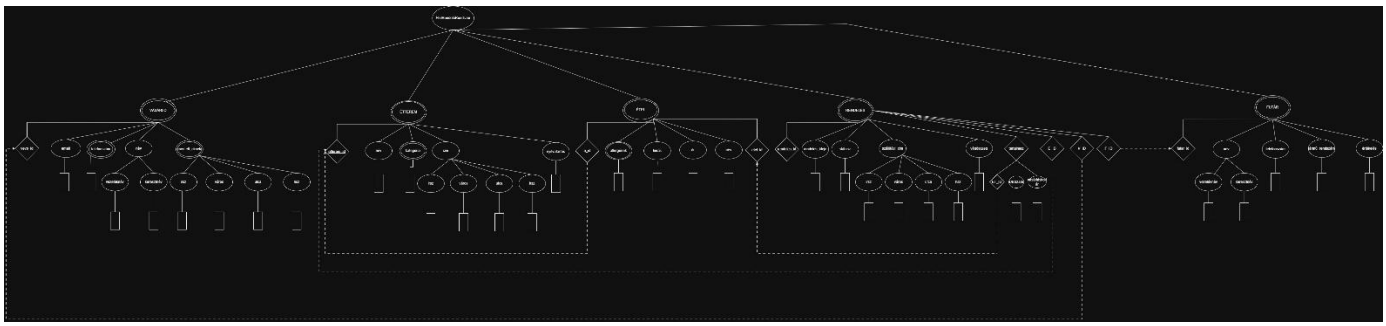
A többértékű tulajdonságok pl. Telefonszam többször ismétlődő gyermekelemekké alakultak.

Kapcsolatok:

1:N kapcsolatok: Az ER modell 1:N kapcsolatait az XDM-ben idegen kulcsok (FK) segítségével modelleztem. Az "N" oldali elemen, például Étel elhelyeztem egy attribútumot (e_id), ami az "1" oldali Étterem etterem_id kulcsára hivatkozik.

M:N kapcsolat: Az ER modell “tartalmaz” kapcsolatát XDM-ben beágyazással oldottam meg. A Rendelés elemen belül létrehoztam egy RendelesTetelei elemet, ami a kapcsolategyednek felel meg. Ezen belül helyeztem el a Tetel elemeket, amelyek tartalmazzák a kapcsolat saját tulajdonságait (pl. Darabszam) és egy IDREF hivatkozást a másik kapcsolódó egyedre (etel_idre).

A modell elkészítése során nagyon figyeltem a feladatkiírás azon feltételére, hogy az elemeket összekötő PK és FK vonalak ne keresztezzék egymást, így biztosítva az ábra átláthatóságát.



2. ábra: XDM modell (Fájl: GH5OGN_XDM.jpg)

4. XML dokumentum

A 3-as pontban megtervezett XDM struktúra alapján elkészítettem a konkrét adatokat tartalmazó GH5OGN_XML.xml dokumentumot. Ez a fájl képezi az adatbázis tényleges példányát. A kód fő elemei:

A dokumentum egyetlen gyökérelemmel rendelkezik, ami az EtelRendeloRendszer. Az xmlns:xsi névtér és az xsi:noNamespaceSchemaLocation="GH5OGN_XMLSchema.xsd" attribútum hozzáadásával összekapcsoltam az XML-t a hozzá tartozó sémával, így bármely XML-értelmező képes validálni a tartalmát a dokumentumnak.

Szigorúan követtem az XDM-ben definiált fa hierarchiát. A fő egyedek, mint a Vasarlo, Etterem a nekik megfelelő gyűjtőelemeken pl. Vasarlok, Ettermek belül helyezkednek el. Minden, többszörösen előforduló egyedtípusból legalább két példányt hoztam létre (2 Vasarlo, 2 Etterem, 2 Futar, 4 Etel, 2 Rendeles).

A modellben többértékűnek jelölt tulajdonságokat (pl. Telefonszam, Kategoria, Allergenek) egyszerűen az elem megismétlésével valósítottuk meg, ahogy az XML-ben szokásos, pl. V001 vásárlónak két <Telefonszam> eleme van.

Az XML kódot a kiírásnak megfelelően megjegyzésekkel láttam el az olvashatóság és az egyes blokkok azonosítása érdekében.

a dokumentum legizgalmasabb része a <Rendeles> elem. Ez az elem a rendszer központi magja, itt fut össze a legtöbb kapcsolat.

Az alábbi R001 azonosítójú rendelés tökéletesen mutatja a modellt:

<Rendeles rendeles_id="R001" vevo_idref="V001" etterem_idref="E001" futar_idref="F001">

<RendelesIdeje>2025-10-27T19:30:00</RendelesIdeje>

<Statusz>Kiszállítva</Statusz>

<SzallitasiCim>

<Irsz>3530</Irsz>

```
      <Varos>Miskolc</Varos>
      <Utca>Szent István utca</Utca>
      <Hsz>1</Hsz>
   </SzallitasiCim>
   <Vegosszeg>3200</Vegosszeg>
   <RendelesTetelei>
      <Tetel etel_idref="ET001">
         <Darabszam>1</Darabszam>
         <VasarlaskoriAr>1400</VasarlaskoriAr>
         <ExtraKeresek>Extra sajttal</ExtraKeresek>
      </Tetel>
      <Tetel etel_idref="ET002">
         <Darabszam>1</Darabszam>
         <VasarlaskoriAr>1800</VasarlaskoriAr>
      </Tetel>
   </RendelesTetelei>
</Rendeles>
```

Kódmagyarázat:

Az elem rendeles_id="R001" elsődleges kulcsa mellett három idegen kulcsot is tartalmaz attribútumként: vevo_idref="V001" (ez azt mutatja ki rendelte), etterem_idref="E001" (honnan rendelte), futar_idref="F001" (ki szállította). Ezek az IDREF-ek kötik össze a tranzakciót a többi fő egyeddel.

Az ER modell tartalmaz kapcsolategyedét ez valósítja meg.

A Tétel maga is rendelkezik tulajdonságokkal (Darabszam, VasarlaskoriAr, ExtraKeresek), ahogy azt az ER modellben meghatároztam, például a darabszám 1, a vásárláskori ár 1400 Ft és extra sajttal kéri a vásárló. Emellett tartalmazza az etel_idref hivatkozást, ami az Etel egyedre mutat.

Több-a-többhöz viszony is látható, mivel ez az egy Rendeles (R001) két különböző Tételt is tartalmaz (ET001 és ET002).

A teljes dokumentum a GH5OGN_XML.xml fájlban található.

5. XMLSchema dokumentum

Az XML Schema (GH5OGN_XMLSchema.xsd) célja, hogy kikényszerítse az előző pontban létrehozott XML dokumentum strukturális szabályait. A séma fogja garantálni, hogy csak az XDM modellünknek megfelelő, érvényes adatok kerültek az adatbázisba.

Az újrafelhasználható struktúrákat xs:complexType-ként definiáltam. Létrehoztam NevTipus , CimTipus, EtteremTipus, VasarTipus stb. típusokat. Ennek nagy előnye az újrafelhasználhatóság: például a CimTipus-t mind a MentettCimek, mind a SzallitasiCim használja.

Az elemek előfordulását minOccurs és maxOccurs attribútumokkal szabályoztam. A többértékű tulajdonságok (pl. Telefonszam, Kategoria , Allergenek) maxOccurs="unbounded" értéket kaptak. Az opcionális elemek, mint a Leiras, ExtraKeresek, minOccurs="0" beállítással rendelkeznek.

A gyökérelem definícióján belül xs:key elemekkel definiáltam az összes egyed elsődleges kulcsát (pl. VasarPK, EtteremPK stb.). Az xpath attribútummal megadtam, hogy melyik elemeken és melyik attribútum/elem képezi a kulcsot.

Aztán az egyik legfontosabb eszköz, az xs:keyref elemek következtek. Ezek biztosítják, hogy minden ..._idref attribútum egy korábban definiált xs:key-re hivatkozzon.

A séma legizgalmasabb része az M:N kapcsolat (Rendelés-Étel) validálása. Ez két részből áll, az EtelPK, ami az étel kulcsa,és a TetelEtelFK, ami a rendelési tétel hivatkozása az ételre.

```
<xs:element name="EtelRendeloRendszer">
  <xs:complexType>
    </xs:complexType>

  <xs:key name="EtelPK">
    <xs:selector xpath="Etelek/Etel"/>
    <xs:field xpath="@etel_id"/>
  </xs:key>

  <xs:keyref name="TetelEtelFK" refer="EtelPK">
    <xs:selector xpath="Rendelesek/Rendeles/RendelesTetelei/Tetel"/>
    <xs:field xpath="@etel_idref"/>
```

```
</xs:keyref>


</xs:element>
```

Kódmagyarázat:

EtelPK: Garantálja, hogy minden Etel egyedi etel_id-val rendelkezzen.

refer="EtelPK": Megmondja, hogy az EtelPK-ban definiált kulcslistához kell referrálni.

selector: Az xpath kifejezés megkeresi az összes létező <Tetel> elemet, bárhol is legyenek beágyazva a Rendelesek-ben.

field: Megvizsgálja az adott <Tetel> elem @etel_idref attribútumát.

Ez ellenőrzi, hogy a etel_idref értéke szerepel-e az EtelPK által generált istában. Ha egy rendelésben olyan etel_idref szerepel, ami nem létezik az <Etelek> listában, a validáció sikertelen lesz. Ezzel garantálható, hogy nem létező ételt nem lehet rendelni.

A teljes séma a GH5OGN_XMLSchema.xsd fájlban található.

6. DOM beolvasás

A feladat második, programozási része egy Java alkalmazás (GH5OGNDOMParse) elkészítése volt, ami a hu.domparse.gh5ogn csomagban helyezkedik el. A feladatkiírásnak megfelelően ez az alkalmazás a DOM API-t használja az első részben létrehozott GH5OGN_XML.xml fájl olvasására, lekérdezésére, változtatására. A GH5OGNDomParse.java-t egy egyszerű menüvel láttam el a főosztályban, amit futtatva meg lehet hívni a három fő funkciót ellátó osztályokat.

Project name: GH5OGNDOMParse

Package: hu.domparse.gh5ogn

Class names: GH5OGNDomRead, GH5OGNDomQuery, GH5OGNDOMModify

Az adatolvasó osztály (GH5OGNDomRead) célja, hogy megcsinálja az XML dokumentum teljes tartalmának beolvasását és feldolgozását. A program nem csupán beolvassa, hanem a struktúrát követve, strukturált és bárki által olvasható formában írja ki az adatokat a konzolra, és egy külső GH5OGN_Read_mentett.txt fájlba is menti az eredményt.

A megvalósításom a JAXP komponenseit használja.

Egy DocumentBuilderFactory és egy DocumentBuilder segítségével a program beolvassa a GH5OGN_XML.xml fájlt, amely egy DOM faként jön létre . A doc.getDocumentElement().normalize() hívás eltávolítja a felesleges üres szövegeket például, biztosítva a tiszta feldolgozást.

A program egyedtípusonként be a fát. A doc.getElementsByTagName("Vasarlo") metódus például egy NodeList-et ad vissza az összes vásárló csomóponttal.

Egy for ciklus megy végig a NodeList-en. Minden Node-ot Element-té kasztol, ami után már használni lehet az element specifikus metódusokat.

element.getAttribute("vevo_id") az attribútumok olvasására.

Vannak függvények, mint a getText(element, "Email") a beágyazott elemek tartalmának kinyerésére.

element.getElementsByTagName("Telefonszam") a többértékű tulajdonságok listájának lekérésére, ezeken egy belső ciklussal megy végig a kód.

A program egy StringWriter-be és egy PrintWriter-be írja a formázott szöveget. Igy ugyanaz a kimenet íródik ki a konzolra System.out.print(buffer.toString())-al és kerül mentésre a fájlba is FileWriter-rel.

```
// Rendelések
NodeList rendelesek = doc.getElementsByTagName("Rendeles");
out.println("Rendelések (" + rendelesek.getLength() + ") \n");
for (int i = 0; i < rendelesek.getLength(); i++) {
  Element rendeles = (Element) rendelesek.item(i);

  // FK-k olvasása
  out.println("[Rendelés] rendeles_id=" + rendeles.getAttribute("rendeles_id")
      + ", vevo_idref=" + rendeles.getAttribute("vevo_idref")
      + ", etterem_idref=" + rendeles.getAttribute("etterem_idref")
      + ", futar_idref=" + rendeles.getAttribute("futar_idref"));

  // Egyszerű elemek olvasása
  out.println("  Időpont: " + getText(rendeles, "RendelesIdeje"));
  out.println("  Státusz: " + getText(rendeles, "Statusz"));

  // Összetett elem olvasása
```

```
    Element szc = (Element) rendeles.getElementsByTagName("SzallitasiCim").item(0);
    out.println("  Szállítási cím: " + formatCim(szc)); // formatCim segédfüggvény
    out.println("  Végösszeg: " + getText(rendeles, "Vegosszeg"));

    // lista feldolgozása
    Element tetelek = (Element) rendeles.getElementsByTagName("RendelesTetelei").item(0);
    if (tetelek != null) {
      NodeList tetelList = tetelek.getElementsByTagName("Tetel");
      // Belső ciklus a tételek bejárására
      for (int t = 0; t < tetelList.getLength(); t++) {
        Element tetel = (Element) tetelList.item(t);
        out.println("    [Tétel] etel_idref=" + tetel.getAttribute("etel_idref"));
        out.println("     Darabszám: " + getText(tetel, "Darabszam"));
        out.println("     Vásárláskori ár: " + getText(tetel, "VasarlaskoriAr"));

        // Opcionális elem kezelése
        String extra = getOptionalText(tetel, "ExtraKeresek");
        if (extra != null) out.println("     Extra kérések: " + extra);
      }
    }
    out.println();
  }
```

Kódmagyarázat

getAttribute():olvassa a <Rendeles> elem attribútumait, az idegen kulcsokat

getText(): Egy saját függvény, ami egyszerűsíti a rendeles.getElementsByTagName("Statusz").item(0).getTextContent() bonyolultságú hívásokat.

formatCim(): Egy másik függvény, ami megkeresi a <SzallitasiCim> elemet, kinyeri annak elemeit (Irsz, Varos), és egy String-é fűzi össze.

A program lekéri a <RendelesTetelei> elemet, majd azon belül az összes <Tetel> elemet, és egy for ciklussal dolgozza fel azokat.

A getOptionalText ellenőrzi, hogy az adott elem (ExtraKeresek) egyáltalán létezik-e, elkerülve ezzel az errort.

A teljes kód a GH5OGNDomRead.java fájlban található.

7. DOM lekérdezés

Az adatlekérdező osztály célja, hogy bemutassa, hogyan lehet információkat kinyerni a DOM fából. Kikötés volt, hogy XPath kifejezéseket nem használhatunk. A lekérdezéseket kizárólag a DOM API alapvető metódusaival valósítottam meg.

A program a DOM fa beolvasása után 5 különböző lekérdezést hajt végre:

Összes vásárló neve és e-mail címe: Egyszerű adatgyűjtés, végig megy az összes Vasarlo elemen.

'E001' étterem ételei: Végig megy az összes Etel elemen, és az etterem_idref attribútum értékének ellenőrzése .equals("E001")

4000 Ft feletti rendelések: Végig megy a Rendeles elemeken, a Vegosszeg tartalmának kinyerése, Integerré alakítása Integer.parseInt() függvénnyel, és összehasonlítása > 4000.

Futárok telefonszáma és értékelése: adatgyűjtés Futar elemekből.

Rendelések tételeinek összesített darabszáma: Egy külső ciklus iterál végig a rendeléseken, egy belső pedig az adott rendelés Tetel elemein, összegezve a Darabszam értékeket egy sum változóba.

```
//2. E001 étteremhez tartozó ételek neve és ára
System.out.println("\n E001 étterem ételeinek neve és ára:");
NodeList etelek = doc.getElementsByTagName("Etel");
for (int i = 0; i < etelek.getLength(); i++) {
   Element etel = (Element) etelek.item(i);

   // Attribútum értékének ellenőrzése
   if ("E001".equals(etel.getAttribute("etterem_idref"))) {
      System.out.println("- " + text(etel, "Nev") + " | " + text(etel, "Ar") + " Ft");
   }
}
```

```
//4000 Ft feletti rendelések azonosítója és státusza
System.out.println("\n 4000 Ft feletti rendelések:");
NodeList rendelesek = doc.getElementsByTagName("Rendeles");
for (int i = 0; i < rendelesek.getLength(); i++) {
   Element r = (Element) rendelesek.item(i);

   int vegosszeg = Integer.parseInt(text(r, "Vegosszeg"));

   // Elem tartalmának ellenőrzése
   if (vegosszeg > 4000) {
      System.out.println("- " + r.getAttribute("rendeles_id") + ": " + vegosszeg + " Ft | " +
text(r, "Statusz"));
   }
}
```

Kódmagyarázat:

getElementsByTagName("Etel") az összes ételt visszaadja. Az if feltétel az, ami elvégzi a szűrést, az etel.getAttribute("etterem_idref") metódusssal lekéri az idegen kulcsot, és egy összehasonlítással (.equals()) dönti el, hogy az adott elem megfelel-e a kritériumnak.

Hasonlóan, az összes Rendeles elemet lekérjük. A szűrés itt összetettebb:

text(r, "Vegosszeg") kinyeri a tartalmat pl. 3200.

Integer.parseInt() átalakítja ezt a Stringet int típusuvá.

Az if (vegosszeg > 4000) végzi a feltételes szűrést.

A teljes kód a GH5OGNDomQuery.java fájlban található.

8. DOM Adatmódosítás

A program a kiírásnak megfelelően 5 különböző módosítást hajt végre.

A program beolvassa a GH5OGN_XML.xml fájlt egy Document objektumba.

A program végigmegy manuálisan az elemlistákon: doc.getElementsByTagName("Vasarlo")

Egy külső ciklus megkeresi a módosításra váró fő elemet pl. a "V002" ID-jű Vasarlot az attribútuma alapján.

Egy belső ciklus getChildNodes() segítségével megy végig a megtalált elem gyerekein.

A node.getNodeType() == Node.ELEMENT_NODE ellenőrzéssel kiszűri a szöveges és egyéb node-okat, aztán a node.getNodeName() alapján azonosítja a módosítandó al elemet, mint pl. a Nyitvatartas.

A módosítások a doc.createElement(), element.appendChild(), element.removeChild() és element.setTextContent() DOM-metódusokkal történnek.

a Transformer osztályt használom, hogy a módosított Document fát egy azaz közvetlenül a konzolra írjam.

A kódrészlet két módosítást mutat, egy meglévő elem tartalmának módosítását, és egy teljesen új elem felépítését.

```
// fa beolvasása
Document doc = builder.parse(xmlFile);
doc.getDocumentElement().normalize();

// 2. MÓDOSÍTÁS: E002 étterem nyitvatartás módosítása
NodeList etteremList = doc.getElementsByTagName("Etterem");
for (int i = 0; i < etteremList.getLength(); i++) {
  Node nNode = etteremList.item(i);
  if (nNode.getNodeType() == Node.ELEMENT_NODE) {
    Element etterem = (Element) nNode;

    // Az "E002" azonosítójú étterem keresése
    if ("E002".equals(etterem.getAttribute("etterem_id"))) {
      NodeList eChildren = etterem.getChildNodes();

      for (int j = 0; j < eChildren.getLength(); j++) {
        Node eNode = eChildren.item(j);
```

```
            if (eNode.getNodeType() == Node.ELEMENT_NODE) {
                // A "Nyitvatartas" elem keresése név alapján
                if ("Nyitvatartas".equals(eNode.getNodeName())) {
                    eNode.setTextContent("H-V: 11:00-23:00");
                    System.out.println("E002 nyitvatartása frissítve");
                    break; // Belső ciklus megszakítása
                }
            }
        }
        break; // Külső ciklus megszakítása
    }
  }
}

// 4. MÓDOSÍTÁS: Új étel felvétele az E001 étteremhez
Node etelek = doc.getElementsByTagName("Etelek").item(0);

// Új Etel elem  felépítése
Element ujEtel = doc.createElement("Etel");
ujEtel.setAttribute("etel_id", "ET005");
ujEtel.setAttribute("etterem_idref", "E001");

Element nev = doc.createElement("Nev");
nev.setTextContent("Retro lángos");
ujEtel.appendChild(nev);

Element ar = doc.createElement("Ar");
ar.setTextContent("1500");
```

```
ujEtel.appendChild(ar);


Element leiras = doc.createElement("Leiras");
leiras.setTextContent("Fokhagyma, tejföl, sajt");
ujEtel.appendChild(leiras);


// A felépített elem hozzáadása a fához
etelek.appendChild(ujEtel);
System.out.println("Új étel felvéve az E001-hez");
...
...
// Módosított XML kiírása a konzolra
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
transformer.setOutputProperty(OutputKeys.INDENT, "yes");
DOMSource source = new DOMSource(doc);


System.out.println("\n---Módosított dokumentum---");
StreamResult consoleResult = new StreamResult(System.out);
transformer.transform(source, consoleResult);
```

Kódmagyarázat:

Először a külső ciklus megkeresi a E002 Éttermet. Ezután egy belső ciklus veszi át az irányítást, ami az étterem összes gyerekén végigmegy. Az if (eNode.getNodeType() == Node.ELEMENT_NODE) sor kiszűri az üres szöveg nodeokat. Csak ezután keresi meg név eNode.getNodeName() alapján a Nyitvatartas elemet, és írja át a tartalmát.

ujEtel elem felépítése lépésről lépésre történik, úgy, hogy doc.createElement("Nev") létrehozza a <Nev> elemet, nev.setTextContent(...) beállítja a tartalmát, és ujEtel.appendChild(nev) hozzáadja az <Etel> elemhez.

new StreamResult(System.out) kerül felhasználásra, ami a teljes módosított XML-t a konzolra írja.

A teljes kód a GH5OGNDOMModify.java fájlban található