Jegyzőkönyv

Webes adatkezelő környezetek

Féléves feladat

A feladat címe

Készítette: Blaskó Balázs Levente

Neptunkód: RQLZPK

Dátum: 2024.12.09.

Bevezetés, feladat leírás:

Ez a feladat egy orvosi rendelő adatkezelési rendszert modellez, amely XML (Extensible Markup Language) formátumban tárolja az adatokat, és azok validálásához XSD (XML Schema Definition) fájlt használ. A rendszer tartalmazza az orvosok, páciensek, vizsgálatok, időpontok és gyógyszerek adatait, amelyeket az XML fájl különböző elemekben struktúráltan tárol. A fájlban szereplő információk például orvosok neve, szakterülete, kezelési időszakai, páciensek adatai, vizsgálatok típusai és dátumai, valamint a gyógyszerek adagolási előírásai és mellékhatásai.

A feladat része, hogy a rendszert XSD séma segítségével validáljuk, amely biztosítja, hogy az XML fájl megfeleljen az előírt struktúrának és típusoknak. Az XSD fájlban meghatározott típusok és elemek segítenek az adatellenőrzésben, biztosítva ezzel, hogy az adatokat helyesen és következetesen kezeljük.

A Java programozásban a DOM (Document Object Model) használata szükséges az XML fájl feldolgozásához. A DOM lehetővé teszi az XML adatok hierarchikus struktúrájának bejárását és azok manipulálását. A DOMParser osztály segítségével az XML fájl beolvasható, és az adatokat programozott módon lekérdezhetjük és feldolgozhatjuk.

Az ER (Entitás-Kapcsolat) modell segítségével az adatbázis logikai struktúráját is ábrázolhatjuk. Az ER diagram tartalmazza az adatokat (például orvosok, páciensek, vizsgálatok), valamint az entitások közötti kapcsolatokat, mint például, hogy mely orvosok kezelik a pácienseket, illetve mely páciensek vesznek részt vizsgálatokon.

Ez a feladat integráltan kezeli az adatokat XML, XSD és DOM technológiák segítségével, miközben az ER modell biztosítja az adatbázis-tervezés logikai struktúráját.

1. Feladat (ER modell, XDM modell, XML, XSD)

1.1 Az adatbázis ER modell tervezése:

Megvalósítás:

Ez a rendszer egy adatbázis alapú egészségügyi információs rendszer modelljét mutatja. A tárolt adatok logikai struktúrája lehetővé teszi:

- Páciensek adatainak kezelését: személyes adatok, vizsgálatok, kezelések nyilvántartása.
- Időpontok nyilvántartását: orvosok és kezelések időbeli ütemezése.
- Gyógyszerek követését: gyógyszeres kezelések és mellékhatások rögzítését.

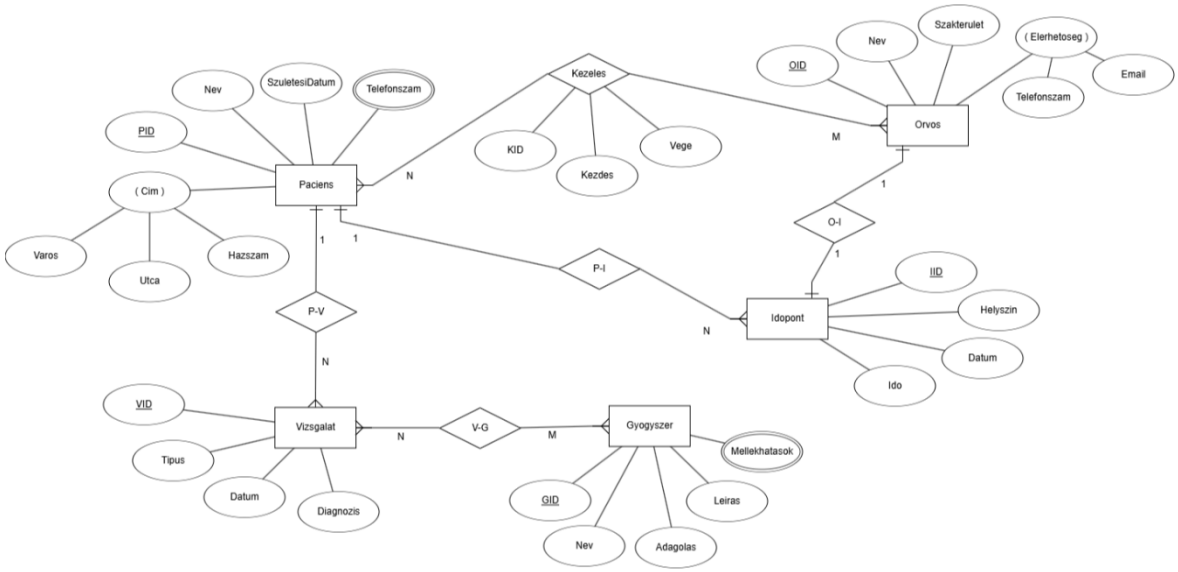
Entitások és attribútumok:

1. Paciensek (Páciensek):
   - Attribútumok: PID (azonosító), Név, Születési dátum, Telefonszám, Cím (Város, Utca, Házszám).
   - Kapcsolat: Kapcsolódik a vizsgálatokhoz (P-V) és kezelésekhez (P-I).
2. Vizsgálatok:
   - Attribútumok: VID (azonosító), Típus, Dátum, Diagnózis.
   - Kapcsolat: Több vizsgálat kapcsolódhat egy pácienshez (P-V). Kapcsolódik gyógyszerekhez is (V-G).
3. Gyógyszerek:
   - Attribútumok: GID (azonosító), Név, Leírás, Adagolás, Mellékhatások.
   - Kapcsolat: Több gyógyszer is kapcsolódhat egy vizsgálathoz (V-G).
4. Kezelések:
   - Attribútumok: KID (azonosító), Kezdés, Vége.
   - Kapcsolat: Egy pácienshez több kezelés tartozhat (P-I).
5. Időpontok:
   - Attribútumok: IID (azonosító), Helyszín, Dátum, Idő.
   - Kapcsolat: Kapcsolódik orvoshoz (O-I).
6. Orvosok:
   - Attribútumok: OID (azonosító), Név, Szakterület, Elérhetőség (Telefonszám, E-mail).
   - Kapcsolat: Egy orvos több időponthoz is kapcsolódhat (O-I).

Kapcsolatok:

- P-V (Páciens-Vizsgálat): Egy páciens több vizsgálatot is elvégeztethet.

- V-G (Vizsgálat-Gyógyszer): Egy vizsgálathoz több gyógyszer rendelhető.
- P-I (Páciens-Kezelés): Egy páciens hez több kezelés is tartozhat.
- O-I (Orvos-Időpont): Egy orvos több időpontot is kezelhet.



1.2 Az adatbázis konvertálása XDM modellre:

Ez az ábra egy hierarchikus struktúrával rendelkező adatmodell, amely az egészségügyi rendszerek működéséhez kapcsolódik. Az entitások és kapcsolataik egy rendelő információs rendszerében használatos adatok logikai struktúráját jelenítik meg.

Főbb elemek:

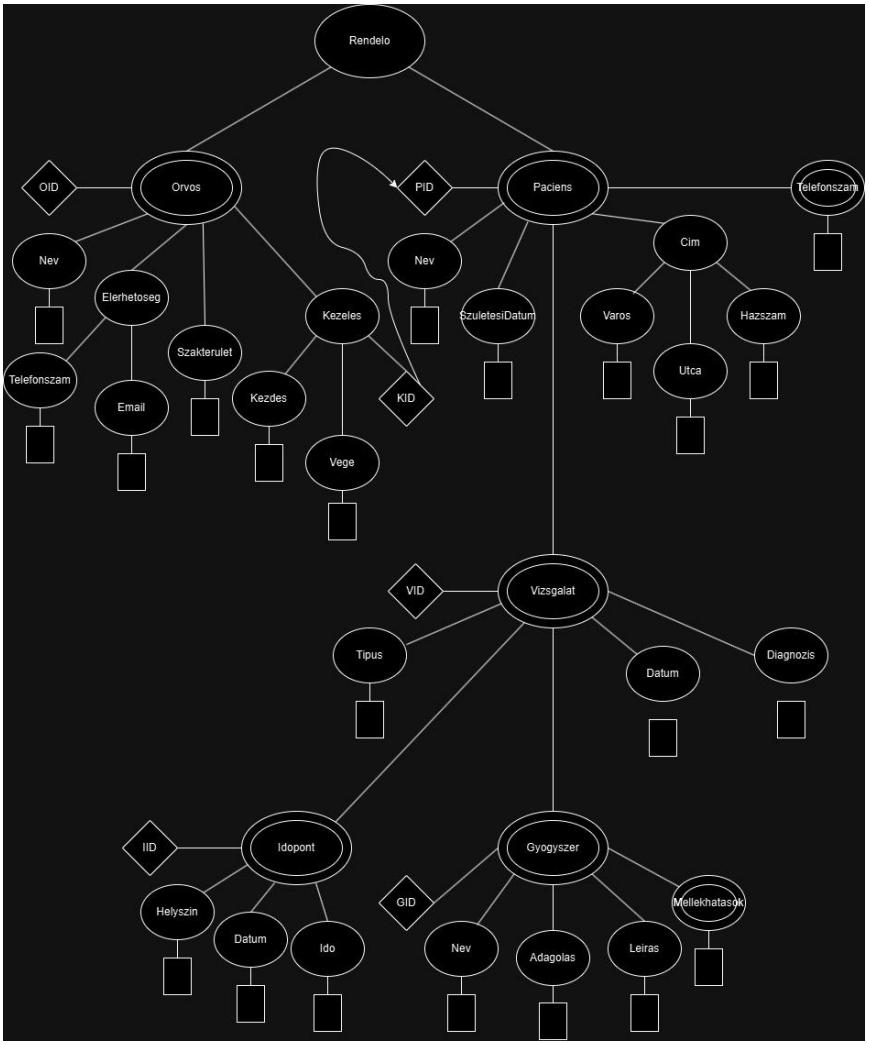
1. **Rendelő:**
   - Az ábra központi eleme, amely köré a többi entitás szerveződik.
2. **Orvos:**
   - Attribútumok: OID (azonosító), Név, Szakterület, Telefonszám, E-mail (elérhetőség).
   - Kapcsolódik a kezelésekhez és időpontokhoz.
3. **Páciensek:**
   - Attribútumok: PID (azonosító), Név, Telefonszám, Születési dátum, Cím (Város, Utca, Házszám).
   - Kapcsolódik a vizsgálatokhoz és kezelésekhez.
4. **Kezelések:**
   - Attribútumok: KID (azonosító), Kezdés, Vége.
   - Kapcsolódik orvoshoz és pácienshez.
5. **Vizsgálatok:**

- Attribútumok: VID (azonosító), Típus, Dátum, Diagnózis.
- Kapcsolódik páciensekhez és gyógyszerekhez.

6. **Időpontok:**
   - Attribútumok: IID (azonosító), Helyszín, Dátum, Idő.
   - Kapcsolódik orvoshoz.
7. **Gyógyszerek:**
   - Attribútumok: GID (azonosító), Név, Adagolás, Leírás, Mellékhatások.
   - Kapcsolódik vizsgálatokhoz.

Kapcsolatok:

- **Orvos és Kezelések:** Egy orvos több kezelésért is felelhet.
- **Páciens és Vizsgálatok:** Egy páciensnek több vizsgálata lehet, amelyek diagnózisokat és gyógyszereket tartalmaznak.
- **Vizsgálatok és Gyógyszerek:** Egy vizsgálat során több gyógyszer is előírható.
- **Időpontok és Orvos:** Az időpontok a rendelői ellátások időbeli szervezéséhez tartoznak.



1.3 Az XDM modell alapján XML dokumentum készítése:

Az XML dokumentum egy egészségügyi rendelő adatainak strukturált leírását tartalmazza, beleértve az orvosok, páciensek, vizsgálatok, időpontok és gyógyszerek részletes adatait. A fájl hierarchikus szerkezete átláthatóvá teszi az adatokat, és lehetővé teszi azok egyszerű feldolgozását egy XML-feldolgozó rendszerben.

Az XML szerkezete

1. Gyökérelem:

A dokumentum gyökéreleme a <Rendelo>, amely tartalmazza az összes alá tartozó adatot.

Az attribútumok közül kiemelendő az xmlns:xsi és az xsi:noNamespaceSchemaLocation, amelyek az XSD séma validációjához szükségesek.

2. Főbb szekciók:
   - <Orvosok>: Az orvosok adatait tartalmazza, beleértve a nevüket, szakterületüket, elérhetőségüket és a kezelésük időtartamát.
   - <Paciensek>: Páciensek részletes adatai, például név, születési dátum, lakcím és telefonszámok.
   - <Vizsgalatok>: A vizsgálatok típusát, dátumát és diagnózisát tartalmazza.
   - <Idopontok>: Időpontok helyszínekkel, dátumokkal és időkkel.
   - <Gyogyszerek>: Gyógyszerek neve, adagolása, leírása és a lehetséges mellékhatások.

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

<Rendelo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="XMLSchemaRQLZPK.xsd">

  <!-- Az alábbiakban négy orvos adatai következnek -->

  <Orvosok>

  <Orvos OID="01">

    <Nev>Dr. Kis László</Nev>

    <Szakterulet>Kardiológia</Szakterulet>

    <Elerhetoseg>

      <Telefon>+36/20 123 4567</Telefon>

      <Email>kislaszlo@gmail.com</Email>

    </Elerhetoseg>

    <Kezeles KID="01">

      <Kezdes>2025-01-01</Kezdes>
```

```
        <Vege>2025-06-01</Vege>
    </Kezeles>
</Orvos>
<Orvos OID="02">
    <Nev>Dr. Kovács Péter</Nev>
    <Szakterulet>Neurológia</Szakterulet>
    <Elerhetoseg>
        <Telefon>+36/30 321 6547</Telefon>
        <Email>kovacspeter@gmail.com</Email>
    </Elerhetoseg>
    <Kezeles KID="02">
        <Kezdes>2025-02-01</Kezdes>
        <Vege>2025-07-01</Vege>
    </Kezeles>
</Orvos>
<Orvos OID="03">
    <Nev>Dr. Szabó István</Nev>
    <Szakterulet>Onkológia</Szakterulet>
    <Elerhetoseg>
        <Telefon>+36/50 654 3231</Telefon>
        <Email>szaboistvan@gmail.com</Email>
    </Elerhetoseg>
    <Kezeles KID="03">
        <Kezdes>2025-03-01</Kezdes>
        <Vege>2025-08-01</Vege>
    </Kezeles>
</Orvos>
<Orvos OID="04">
    <Nev>Dr. Lakatos András</Nev>
    <Szakterulet>Ortopédia</Szakterulet>
    <Elerhetoseg>
        <Telefon>+36/70 987 4561</Telefon>
```

```
            <Email>lakatosandras@gmail.com</Email>
        </Elerhetoseg>
        <Kezeles KID="04">
            <Kezdes>2025-04-01</Kezdes>
            <Vege>2025-09-01</Vege>
        </Kezeles>
    </Orvos>
</Orvosok>
    <!-- Az alábbiakban négy páciens adatai következnek -->
    <Paciensek>
        <Paciens PID="01">
            <Nev>Vörös Tibor</Nev>
            <SzuletesiDatum>1991-01-01</SzuletesiDatum>
            <Cim>
                <Varos>Miskolc</Varos>
                <Utca>Petőfi Sándor utca</Utca>
                <Hazszam>1</Hazszam>
            </Cim>
            <Telefonszamok>
                <Telefonszam>+36/20 456 7817</Telefonszam>
                <Telefonszam>+36/30 456 7822</Telefonszam>
            </Telefonszamok>
        </Paciens>
        <Paciens PID="02">
            <Nev>Kovács Nándor</Nev>
            <SzuletesiDatum>1992-01-01</SzuletesiDatum>
            <Cim>
                <Varos>Miskolc</Varos>
                <Utca>Győri kapu</Utca>
                <Hazszam>2</Hazszam>
            </Cim>
            <Telefonszamok>
```

```
            <Telefonszam>+36/20 456 7831</Telefonszam>
        </Telefonszamok>
    </Paciens>
    <Paciens PID="03">
        <Nev>Bence Károly</Nev>
        <SzuletesiDatum>1993-01-01</SzuletesiDatum>
        <Cim>
            <Varos>Felsőzsolca</Varos>
            <Utca>József Attila utca</Utca>
            <Hazszam>3</Hazszam>
        </Cim>
        <Telefonszamok>
            <Telefonszam>+36/20 456 1785</Telefonszam>
            <Telefonszam>+36/70 456 2786</Telefonszam>
        </Telefonszamok>
    </Paciens>
    <Paciens PID="04">
        <Nev>Kertész Péter</Nev>
        <SzuletesiDatum>1994-01-01</SzuletesiDatum>
        <Cim>
            <Varos>Szeged</Varos>
            <Utca>Rákóczi utca</Utca>
            <Hazszam>4</Hazszam>
        </Cim>
        <Telefonszamok>
            <Telefonszam>+36/70 456 7854</Telefonszam>
            <Telefonszam>+36/30 456 7865</Telefonszam>
        </Telefonszamok>
    </Paciens>
</Paciensek>
<!-- Az alábbiakban négy vizsgálat adatai következnek -->
<Vizsgalatok>
```

```
    <Vizsgalat VID="01">
        <Tipus>Laborvizsgálat</Tipus>
        <Datum>2025-01-15</Datum>
        <Diagnozis>Vérkép elemzés</Diagnozis>
    </Vizsgalat>
    <Vizsgalat VID="02">
        <Tipus>Röntgenvizsgálat</Tipus>
        <Datum>2025-02-15</Datum>
        <Diagnozis>Mellkas röntgen</Diagnozis>
    </Vizsgalat>
    <Vizsgalat VID="03">
        <Tipus>Ultrahang vizsgálat</Tipus>
        <Datum>2025-03-15</Datum>
        <Diagnozis>Hasüreg ultrahang</Diagnozis>
    </Vizsgalat>
    <Vizsgalat VID="04">
        <Tipus>Kardiológiai vizsgálat</Tipus>
        <Datum>2025-04-15</Datum>
        <Diagnozis>Ekg vizsgálat</Diagnozis>
    </Vizsgalat>
</Vizsgalatok>
    <!-- Az alábbiakban négy időpont adatai következnek -->
    <Idopontok>
    <Idopont IID="01">
        <Helyszin>MÁV Rendelőintézet</Helyszin>
        <Datum>2024-12-10</Datum>
        <Ido>09:00</Ido>
    </Idopont>
    <Idopont IID="02">
        <Helyszin>BAZ Vármegyei Központi Kórház</Helyszin>
        <Datum>2024-12-12</Datum>
        <Ido>10:30</Ido>
```

```
    </Idopont>
    <Idopont IID="03">
        <Helyszin>MÁV Rendelőintézet</Helyszin>
        <Datum>2024-12-15</Datum>
        <Ido>14:00</Ido>
    </Idopont>
    <Idopont IID="04">
        <Helyszin>BAZ Vármegyei Központi Kórház</Helyszin>
        <Datum>2024-12-18</Datum>
        <Ido>11:45</Ido>
    </Idopont>
</Idopontok>
    <!-- Az alábbiakban négy gyógyszer adatai következnek -->
    <Gyogyszerek>
    <Gyogyszer GID="01">
        <Nev>Ibuprofen</Nev>
        <Adagolas>200-400 mg naponta 3-4 alkalommal</Adagolas>
        <Leiras>Az ibuprofen egy nem szteroid gyulladáscsökkentő gyógyszer (NSAID), amelyet fájdalom és gyulladás csökkentésére használnak.</Leiras>
        <Mellekhatasok>
            <Mellekhatas>Fejfájás, szédülés, gyomorpanaszok, hányinger</Mellekhatas>
        </Mellekhatasok>
    </Gyogyszer>
    <Gyogyszer GID="02">
        <Nev>Aspirin</Nev>
        <Adagolas>75-100 mg naponta egyszer</Adagolas>
        <Leiras>Az aszpirin egy nem szteroid gyulladáscsökkentő gyógyszer, amelyet főként a szív- és érrendszeri betegségek megelőzésére és kezelésére használnak.</Leiras>
        <Mellekhatasok>
            <Mellekhatas>Gyomorirritáció, hányinger, vérzés</Mellekhatas>
            <Mellekhatas>Allergiás reakciók, bőrkiütés</Mellekhatas>
        </Mellekhatasok>
    </Gyogyszer>
```

```
    <Gyogyszer GID="03">
        <Nev>Paracetamol</Nev>
        <Adagolas>500-1000 mg naponta 3-4 alkalommal, maximum 4000 mg/nap</Adagolas>
        <Leiras>Paracetamol egy fájdalomcsillapító és lázcsillapító szer, amelyet enyhe és közepes erősségű fájdalom kezelésére használnak.</Leiras>
        <Mellekhatasok>
            <Mellekhatas>Szédülés, hányinger, bélpanaszok</Mellekhatas>
            <Mellekhatas>Allergiás reakciók, kiütések</Mellekhatas>
            <Mellekhatas>Májműködési zavarok (túladagolás esetén)</Mellekhatas>
        </Mellekhatasok>
    </Gyogyszer>
    <Gyogyszer GID="04">
        <Nev>Amoxicillin</Nev>
        <Adagolas>250-500 mg 8 óránként</Adagolas>
        <Leiras>Az amoxicillin egy széles spektrumú antibiotikum, amelyet bakteriális fertőzések kezelésére használnak.</Leiras>
        <Mellekhatasok>
            <Mellekhatas>Hasmenés, hányinger, bélflóra felborulás</Mellekhatas>
            <Mellekhatas>Allergiás reakciók, bőrkiütés</Mellekhatas>
            <Mellekhatas>Gombás fertőzések (pl. szájpenész)</Mellekhatas>
            <Mellekhatas>Vérképzőszervi elváltozások, mint pl. leukopenia</Mellekhatas>
        </Mellekhatasok>
    </Gyogyszer>
</Gyogyszerek>
</Rendelo>
```

1.4 Az XML dokumentum alapján XMLSchema készítése:

XML-séma az XML dokumentumok szerkezetének és tartalmának szabványos meghatározását biztosítja. Az alábbi XML-séma az egészségügyi rendelő rendszeréhez készült, amely az alábbi fő elemeket tartalmazza:

- **Rendelo**: A gyökérelem, amely összefogja az összes alapelemet (Orvosok, Paciensek, Vizsgálatok, Időpontok, Gyógyszerek).

- **Orvosok**: Az orvosok adatait tárolja, beleértve a nevüket, szakterületüket, elérhetőségüket és a kezeléshez tartozó információkat. Az **OID** attribútum egyedi azonosítót biztosít minden orvos számára.
- **Paciensek**: A páciensek adatai, beleértve a nevüket, születési dátumukat, címüket és telefonszámaikat. Az **PID** attribútum biztosítja az egyediséget.
- **Vizsgálatok**: A vizsgálatok típusát, dátumát és diagnózisát írja le. Az **VID** attribútum biztosítja az azonosítást.
- **Idopontok**: Az időpontok helyszínét, dátumát és időpontját tárolja. Az **IID** attribútum biztosítja az egyediséget.
- **Gyogyszerek**: A gyógyszerek nevét, adagolását, leírását és mellékhatásait tartalmazza. Az **GID** attribútum segítségével azonosíthatók.

Speciális típusok és kulcsok:

- **Egyedi azonosítók**: Az összes fő entitás esetén biztosított kulcsok (pl. OID, PID, stb.) garantálják az adatok egyediségét.
- **Hivatkozások (keyref)**: Az XML-séma biztosítja az adatok közötti kapcsolatokat, pl. egy kezelés egy adott pácienshez kapcsolható.

Egyszerű típusok:

- **PhoneNumber**: Telefonszámok érvényesítésére (pl. +36/30 123 4567 formátum).
- **Appointment**: Időpontok érvényesítése (HH:mm formátum).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- Az alábbiakban egyedi típusok definiálása -->
  <xs:element name="Rendelo">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Orvosok">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Orvos" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Nev" type="xs:string"/>
```

```
                    <xs:element name="Szakterulet" type="xs:string"/>
                    <xs:element name="Elerhetoseg">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Telefon" type="PhoneNumber"/>
                                <xs:element name="Email" type="xs:string"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="Kezeles" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Kezdes" type="xs:date"/>
                                <xs:element name="Vege" type="xs:date"/>
                            </xs:sequence>
                            <xs:attribute name="KID" type="xs:int" use="required"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="OID" type="xs:int" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Paciensek">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Paciens" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Nev" type="xs:string"/>
```

```
                    <xs:element name="SzuletesiDatum" type="xs:date"/>
                    <xs:element name="Cim">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Varos" type="xs:string"/>
                                <xs:element name="Utca" type="xs:string"/>
                                <xs:element name="Hazszam" type="xs:int"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="Telefonszamok">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Telefonszam" type="PhoneNumber" maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="PID" type="xs:int" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Vizsgalatok">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Vizsgalat" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Tipus" type="xs:string"/>
                        <xs:element name="Datum" type="xs:date"/>
```

```
                    <xs:element name="Diagnozis" type="xs:string"/>
                </xs:sequence>
                <xs:attribute name="VID" type="xs:int" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Idopontok">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Idopont" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Helyszin" type="xs:string"/>
            <xs:element name="Datum" type="xs:date"/>
            <xs:element name="Ido" type="Appointment"/>
          </xs:sequence>
          <xs:attribute name="IID" type="xs:int" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Gyogyszerek">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Gyogyszer" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Nev" type="xs:string"/>
            <xs:element name="Adagolas" type="xs:string"/>
```

```
                        <xs:element name="Leiras" type="xs:string"/>
                        <xs:element name="Mellekhatasok">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Mellekhatas" type="xs:string" maxOccurs="unbounded"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                    <xs:attribute name="GID" type="xs:int" use="required"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:key name="Orvos_key">
    <xs:selector xpath="Orvos"/>
    <xs:field xpath="@OID"/>
</xs:key>
<xs:key name="Paciens_key">
    <xs:selector xpath="Paciens"/>
    <xs:field xpath="@PID"/>
</xs:key>
<xs:key name="Vizsgalat_key">
    <xs:selector xpath="Vizsgalat"/>
    <xs:field xpath="@VID"/>
</xs:key>
<xs:key name="Idopont_key">
    <xs:selector xpath="Idopont"/>
    <xs:field xpath="@IID"/>
```

```
        </xs:key>
        <xs:key name="Gyogyszer_key">
            <xs:selector xpath="Gyogyszer"/>
            <xs:field xpath="@GID"/>
        </xs:key>
        <xs:key name="Kezeles_key">
            <xs:selector xpath="Kezeles"/>
            <xs:field xpath="@KID"/>
        </xs:key>
        <xs:keyref name="FK_Kezeles_Paciens" refer="Paciens_key">
            <xs:selector xpath="Kezeles"/>
            <xs:field xpath="@PID"/>
        </xs:keyref>
    </xs:element>
<xs:simpleType name="PhoneNumber">
    <xs:restriction base="xs:string">
        <xs:pattern value="\+36/\d{2} \d{3} \d{4}"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Appointment">
    <xs:restriction base="xs:string">
        <xs:pattern value="\d{2}:\d{2}"/>
    </xs:restriction>
</xs:simpleType>
</xs:schema>
```

2.Feladat (DOMParse)

2.1 adatolvasás:

```
// Attribútumok kiírása (ha vannak)
                    NamedNodeMap attributes = itemElement.getAttributes();
```

```
for (int k = 0; k < attributes.getLength(); k++) {
    Node attribute = attributes.item(k);
    output.append("    Azonosító: ")
        .append(attribute.getNodeName())
        .append(" = ")
        .append(attribute.getNodeValue()).append("\n");
}
```

Ez a kódrészlet egy Java programban használható, és az XML-dokumentum feldolgozásának egy részletét valósítja meg. Kifejezetten az attribútumok kiolvasására és megjelenítésére szolgál egy adott XML-elemhez kapcsolódóan. A következőkben részletesen elmagyarázom a működését és tervezési megfontolásait.

1. **Attribútumok lekérdezése**:

   NamedNodeMap attributes = itemElement.getAttributes();

   Az itemElement egy XML-elem, amely az aktuálisan feldolgozott csomópontot reprezentálja (pl. egy <Orvos> vagy <Paciens> elem). Az getAttributes() metódus visszaad egy NamedNodeMap objektumot, amely tartalmazza az elemhez tartozó összes attribútumot.

2. **Attribútumok iterálása**:

   for (int k = 0; k < attributes.getLength(); k++) {

   Node attribute = attributes.item(k);

   A NamedNodeMap objektum iterálható, és a ciklus az összes attribútumon végighalad. Az item(k) metódus az adott indexű attribútumot adja vissza Node típusban.

3. **Attribútumok kiíratása**:

   output.append("   Azonosító: ")

   .append(attribute.getNodeName())

   .append(" = ")

   .append(attribute.getNodeValue()).append("\n");

   Az attribútum neve az getNodeName() metódussal érhető el (pl. OID, PID).

   Az attribútum értéke az getNodeValue() metódussal szerezhető meg (pl. 123, 45).

Az eredmény a StringBuilder típusú output objektumba kerül (feltételezhetően a kimenet tárolására szolgál).

Tervezési megvalósítás

1. **Modularitás**:

A kódrészlet célzottan az attribútumok kezelésére és kiírására fókuszál, amely egy jól elkülönített feladat. Ez lehetővé teszi a kód egyszerű karbantartását és újrafelhasználhatóságát.

2. **Rugalmasság**:

Az iterálás biztosítja, hogy bármilyen számú attribútum kezelhető legyen, függetlenül attól, hogy az adott XML-elemhez hány attribútum tartozik.

3. **Interoperabilitás**:

A DOM (Document Object Model) API használata lehetővé teszi az XML-adatok szabványos feldolgozását, ami előnyös az interoperabilitás szempontjából. Az API támogatja a legtöbb XML-feldolgozási feladatot.

4. **Érthető kimenet**:

A kimenet formázott és érthető (pl. Azonosító: OID = 123), ami segít a debugolásban vagy a felhasználók számára megjelenített információk áttekintésében.

2.2 adatírás:

```
// Fa struktúra kiíratása rekurzívan
    private static void printNode(Node node, int depth) {
        String indent = "  ".repeat(depth);
        System.out.println(indent + "Név: " + node.getNodeName() + ", Érték: " + node.getNodeValue());
        NamedNodeMap attributes = node.getAttributes();
        if (attributes != null) {
            for (int i = 0; i < attributes.getLength(); i++) {
                Node attribute = attributes.item(i);
                System.out.println(indent + "  Attribútum: " + attribute.getNodeName() + " = " + attribute.getNodeValue());
            }
        }
        NodeList children = node.getChildNodes();
        for (int i = 0; i < children.getLength(); i++) {
```

```
        Node child = children.item(i);
        if (child.getNodeType() == Node.ELEMENT_NODE ||
            (child.getNodeType() == Node.TEXT_NODE && !child.getNodeValue().trim().isEmpty())) {
            printNode(child, depth + 1);
        }
    }
}
```

Ez a kódrészlet egy rekurzív algoritmust valósít meg, amely bejárja és kiírja egy XML-dokumentum **fa-struktúráját**. A módszer bemenetként egy XML-csomópontot (Node) és a mélységi szintet (depth) vár, majd az aktuális csomópont nevét, értékét, attribútumait és gyermekeit jeleníti meg hierarchikus formában.

Kód részletes magyarázata

1. **Behúzás előállítása a mélység alapján**

   String indent = " ".repeat(depth);

   A depth paraméter az aktuális csomópont mélységét jelzi a fában.

   A " ".repeat(depth) segítségével megfelelő számú behúzást generálunk, hogy vizuálisan is látható legyen a hierarchia a kimenetben.

2. **Csomópont neve és értéke kiíratása**

   System.out.println(indent + "Név: " + node.getNodeName() + ", Érték: " + node.getNodeValue());

   node.getNodeName(): Az aktuális csomópont neve (pl. <Orvos>, <Paciens> vagy attribútum esetén az attribútum neve).

   node.getNodeValue(): Az aktuális csomópont értéke. Szövegcsomópontoknál (TEXT_NODE) az érték maga a szöveg, míg elemeknél általában null.

3. **Attribútumok feldolgozása**

   NamedNodeMap attributes = node.getAttributes();

   if (attributes != null) {

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

         Node attribute = attributes.item(i);

```
        System.out.println(indent + "  Attribútum: " + attribute.getNodeName() + " = " +
attribute.getNodeValue());
    }
}
```

Az getAttributes() metódus visszaadja az aktuális csomóponthoz tartozó attribútumokat.

Ha vannak attribútumok, azokat egy for ciklusban dolgozza fel.

Minden attribútum nevét és értékét kiírja egy újabb behúzással.

4. Gyermekcsomópontok feldolgozása

```
NodeList children = node.getChildNodes();
for (int i = 0; i < children.getLength(); i++) {
    Node child = children.item(i);
    if (child.getNodeType() == Node.ELEMENT_NODE ||
        (child.getNodeType() == Node.TEXT_NODE && !child.getNodeValue().trim().isEmpty()))
{
        printNode(child, depth + 1);
    }
}
```

Az getChildNodes() metódus visszaadja az aktuális csomópont gyermekcsomópontjait.

A ciklus minden gyermekcsomópontot feldolgoz, de csak akkor hívja meg rekurzívan a printNode metódust, ha:

A csomópont egy **elemcsomópont** (ELEMENT_NODE, pl. <Orvos>, <Paciens>).

Vagy egy nem üres szövegcsomópont (TEXT_NODE, pl. egy szöveges érték).

A rekurzióval a gyermekcsomópontok mélységét növeli egyel (depth + 1), ami a behúzás változásában is megjelenik.

Kimenet például

Ha az XML-dokumentum így néz ki:

```
<Rendelo>
 <Orvos OID="123">
  <Nev>Dr. Kovács</Nev>
  <Szakterulet>Kardiológia</Szakterulet>
 </Orvos>
</Rendelo>
```

A kód kimenete:

```
Név: Rendelo, Érték: null
 Név: Orvos, Érték: null
  Attribútum: OID = 123
  Név: Nev, Érték: Dr. Kovács
  Név: Szakterulet, Érték: Kardiológia
```

Tervezési megvalósítás

1. **Rekurzív megközelítés**

   A rekurzív hívások biztosítják, hogy az XML-dokumentum **tetszőleges mélységű fáját** fel tudjuk dolgozni.

   A fa szerkezetének bejárása **mélységi keresés** (DFS, Depth-First Search) algoritmus alapján történik.

2. **Moduláris feldolgozás**

   A printNode egy önálló, újrafelhasználható metódus, amely egyetlen feladatra koncentrál: a csomópontok és azok gyermekei kiírására.

3. **Rugalmas attribútumkezelés**

   Az attribútumok kezelését külön kódrészlet valósítja meg, amely biztosítja, hogy az összes attribútum feldolgozásra és megjelenítésre kerüljön.

4. **Szűrés és formázás**

A szűrés biztosítja, hogy üres vagy irreleváns csomópontok (pl. üres szövegcsomópontok) ne jelenjenek meg.

A mélység alapú behúzás javítja a kimenet érthetőségét, különösen bonyolultabb XML-struktúrák esetén.

2.3 adatlekérdezés:

```
// 1. Lekérdezés: Összes "Orvos" elem neve és szakterülete
        System.out.println("\n1. Összes orvos neve és szakterülete:");
        NodeList orvosok = document.getElementsByTagName("Orvos");
        for (int i = 0; i < orvosok.getLength(); i++) {
            Element orvos = (Element) orvosok.item(i);
            String nev = orvos.getElementsByTagName("Nev").item(0).getTextContent();
            String szakterulet = orvos.getElementsByTagName("Szakterulet").item(0).getTextContent();
            System.out.println("Név: " + nev + ", Szakterület: " + szakterulet);
        }
```

Kód részletes magyarázata

Ez a kódrészlet egy XML-dokumentumból lekérdezi az összes <Orvos> elemet, majd kiírja az orvosok nevét és szakterületét.

1. Összes <Orvos> elem kinyerése

NodeList orvosok = document.getElementsByTagName("Orvos");

A document.getElementsByTagName("Orvos") egy NodeList típusú objektumot ad vissza, amely az XML-dokumentumban található összes <Orvos> elemet tartalmazza.

A NodeList egy lista, amelyet index alapján lehet bejárni.

2. Lista bejárása

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

Element orvos = (Element) orvosok.item(i);

A ciklus végigmegy az összes <Orvos> elem listáján.

Az orvosok.item(i) metódussal az aktuális elemet kapjuk meg, amelyet azután Element típusúvá alakítunk, hogy az Element osztály metódusait elérhessük.

3. Adatok lekérdezése

String nev = orvos.getElementsByTagName("Nev").item(0).getTextContent();

String szakterulet = orvos.getElementsByTagName("Szakterulet").item(0).getTextContent();

Név és szakterület lekérdezése:

Az orvos.getElementsByTagName("Nev") az aktuális <Orvos> elem <Nev> gyermekeit keresi meg. Az item(0) az első <Nev> elemet adja vissza.

A getTextContent() az elem tartalmát (a szövegét) adja vissza.

Ugyanígy jár el a <Szakterulet> elemnél, hogy az orvos szakterületét kinyerje.

4. Adatok megjelenítése

System.out.println("Név: " + nev + ", Szakterület: " + szakterulet);

Az orvos neve és szakterülete kerül kiírásra.

Kimenet például

Ha az XML-dokumentumban az alábbi adatok vannak:

```
<Orvos>
   <Nev>Dr. Kovács</Nev>
   <Szakterulet>Kardiológia</Szakterulet>
</Orvos>
<Orvos>
   <Nev>Dr. Tóth</Nev>
   <Szakterulet>Belgyógyászat</Szakterulet>
</Orvos>
```

A kimenet:

1. Összes orvos neve és szakterülete:

Név: Dr. Kovács, Szakterület: Kardiológia

Név: Dr. Tóth, Szakterület: Belgyógyászat

Tervezés és megvalósítás

1. Egyszerű és célzott lekérdezés

A getElementsByTagName() metódus egy gyors megoldás az adott elem összes előfordulásának kinyerésére.

Ez a módszer feltételezi, hogy a <Nev> és <Szakterulet> elemek mindig jelen vannak az <Orvos> elem gyermekeiként. Ha nem, hibakezelés szükséges.

2. Hierarchikus struktúra feldolgozása

A hierarchikus XML-fájl szerkezete alapján a <Nev> és <Szakterulet> elemek az <Orvos> elem közvetlen gyermekei. Ezt a hierarchiát használja ki a lekérdezés.

3. Olvasható kimenet

Az orvosok neve és szakterülete jól strukturált formában kerül megjelenítésre, ami az XML-dokumentumban tárolt adatok egyszerű bemutatására szolgál.

2.4 adatmódosítás:

```
// 4. Módosítás: Egy vizsgálat dátumának frissítése
        System.out.println("\n4. Egy vizsgálat (VID=04) dátumának frissítése:");
        NodeList vizsgalatok = document.getElementsByTagName("Vizsgalat");
        for (int i = 0; i < vizsgalatok.getLength(); i++) {
            Element vizsgalat = (Element) vizsgalatok.item(i);
            if (vizsgalat.getAttribute("VID").equals("04")) {
                        Node tipus = vizsgalat.getElementsByTagName("Tipus").item(0);
                        String vizsgalatTipus = tipus.getTextContent();
                        System.out.println("Vizsgálat típusa: " + vizsgalatTipus);
                Node datum = vizsgalat.getElementsByTagName("Datum").item(0);
                String regiDatum = datum.getTextContent();
                datum.setTextContent("2025-05-01");
                System.out.println("Régi dátum:" + regiDatum);
                System.out.println("Frissített dátum: " + datum.getTextContent());
            }
        }
```

Kód részletes magyarázata

Ez a kódrészlet egy XML-dokumentumból egy adott vizsgálat (VID=04) dátumát frissíti. A frissített adatokat kiírja a konzolra.

1. Összes <Vizsgalat> elem kinyerése

NodeList vizsgalatok = document.getElementsByTagName("Vizsgalat");

A getElementsByTagName("Vizsgalat") metódus az XML-dokumentum összes <Vizsgalat> elemét visszaadja egy NodeList formájában.

Ez egy lista, amelyet index alapján be lehet járni.

2. Ciklus az <Vizsgalat> elemek bejárására

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

Element vizsgalat = (Element) vizsgalatok.item(i);

A ciklus végigmegy az összes <Vizsgalat> elemen.

Az aktuális elemet Element típussá alakítjuk, hogy hozzáférjünk annak attribútumaihoz és gyermekelemeihez.

3. Azonosítás VID alapján

if (vizsgalat.getAttribute("VID").equals("04")) {

Az Element.getAttribute("VID") metódus lekéri az aktuális <Vizsgalat> elem VID attribútumának értékét.

Az attribútum értékét összehasonlítjuk a "04" értékkel. Ha egyezés van, belépünk az if ágba.

4. Vizsgálat típusának kiíratása

Node tipus = vizsgalat.getElementsByTagName("Tipus").item(0);

String vizsgalatTipus = tipus.getTextContent();

System.out.println("Vizsgálat típusa: " + vizsgalatTipus);

A getElementsByTagName("Tipus") metódus az aktuális <Vizsgalat> elem <Tipus> gyermekelemét keresi meg.

Az item(0) hivatkozással az első (és feltételezhetően egyetlen) <Tipus> elemre mutatunk.

A getTextContent() a <Tipus> elem szöveges tartalmát adja vissza, amely a vizsgálat típusa.

5. Régi dátum lekérdezése és kiíratása

Node datum = vizsgalat.getElementsByTagName("Datum").item(0);

String regiDatum = datum.getTextContent();

System.out.println("Régi dátum:" + regiDatum);

Hasonlóan a <Tipus> elemhez, itt a <Datum> elem tartalmát nyerjük ki.

Az item(0) az első <Datum> elemre hivatkozik.

Az getTextContent() a dátum jelenlegi (régi) értékét adja vissza.

6. Dátum frissítése

datum.setTextContent("2025-05-01");

System.out.println("Frissített dátum: " + datum.getTextContent());

Az setTextContent("2025-05-01") metódus segítségével a <Datum> elem tartalmát a "2025-05-01" értékre állítjuk.

Az új dátumot kiíratjuk a konzolra.

Példa XML-részlet előtte

<Vizsgalat VID="04">

<Tipus>Ultrahang</Tipus>

<Datum>2023-12-01</Datum>

<Diagnozis>Normál állapot</Diagnozis>

</Vizsgalat>

4. Egy vizsgálat (VID=04) dátumának frissítése:

Vizsgálat típusa: Ultrahang

Régi dátum: 2023-12-01

Frissített dátum: 2025-05-01

Frissített XML:

<Vizsgalat VID="04">

<Tipus>Ultrahang</Tipus>

<Datum>2025-05-01</Datum>

<Diagnozis>Normál állapot</Diagnozis>

</Vizsgalat>

Tervezés és megvalósítás

1. Adat módosítása XML-ben

Az Element osztály lehetőséget ad arra, hogy közvetlenül módosítsuk az XML-elemek tartalmát (setTextContent() metódus).

Ez a megoldás közvetlen szerkesztés, amely gyors és egyszerű, ha a dokumentum már be van töltve memóriába.

2. Adat azonosítása

A vizsgálatok közül az egyedi VID attribútum alapján történik azonosítás.

Ez egy egyszerű, egyedi kulcson alapuló azonosítás, amely hatékony, ha minden elemnek van egyedi azonosítója.

3. Hibakezelés

Ha a getElementsByTagName("Datum") vagy más hivatkozások nem találják a keresett elemet, akkor NullPointerException fordulhat elő.

Hibakezelést érdemes beépíteni:

```
if (datum != null) {
   datum.setTextContent("2025-05-01");
} else {
   System.out.println("Nem található a vizsgálat dátuma.");
}
```

4. Konzisztens adatfrissítés

A frissített dátum azonnal elérhető az XML-dokumentum memóriába betöltött változatában.

Az XML-dokumentum fizikai fájlban történő mentéséhez szükség van további lépésekre (pl. Transformer osztály használata).

5. Bővíthetőség

A megközelítés könnyen kiterjeszthető más elemek módosítására vagy új attribútumok kezelésére.

Az attribútum-alapú azonosítást más típusú XML-elemeknél is alkalmazhatjuk.