JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Okmányiroda

Készítette: **Potoczki Gitta**

Neptun: **V1C6ND**

Dátum: 2023.12.10

Tartalomjegyzék

Bevezetés ... 4
A feladat leírása: ... 4
1. feladat ... 5
Egyedek: ... 5
Kapcsolatok: ... 6
1b) Az adatbázis konvertálása XDM modellre ... 7
1c) Az XDM modell alapján XML dokumentum készítése ... 7
1d) Az XML dokumentum alapján XMLSchema készítése ... 11
2. feladat ... 15
2a) Adatolvasás ... 15
readDolgozok metódus ... 15
readMunkakorok metódus ... 15
readSzerepek metódus ... 16
readMunkakorIsm metódus ... 16
readAblakok metódus ... 16
readUgyfelek metódus: ... 16
readFelettesek metódus: ... 16
printElement metódus: ... 16
2b) Adatmódosítás ... 21
Dolgozó nevének Módosítása ... 21
Munkakör megnevezésének megváltoztatása ... 21
Munkakör ismeretek módosítása ... 22
Ügykör változtatása ... 22
Ügyfél személyigazolványának módosítása ... 22
Felettes nevének módosítása: ... 22
2c) Adatlekérdezés ... 26
Ügyfelek személyes adatainak lekérése ... 26
Dolgozók szűrése születési dátum alapján ... 26
Elemek ABC sorrendbe rendezése ... 26
Munkakör ismeretek statisztikája ... 26
Ügyfelek látogatásának időtartama ... 26
2d) Adatírás ... 33
XML fájl létrehozása ... 33
Dolgozók, Munkakörök, Szerepek, Munkakör Ismeretek, Ablakok, Ügyfelek, Felettesek elementek létrehozása ... 33

XML tranzformáció és kiíratás 34
A fájlba írás eredménye 41

Bevezetés

A feladat leírása:

A feladat egy okmányiroda forgalmának nyilvántartására épül. Az egyedkapcsolatok minden formája megjelenik: egy-egy, egy-több és több-több kapcsolatok is szerepelnek az okmányiroda modelljében.

A modell 5 egyedet tartalmaz:

- Ugyfel
- Ablak
- Dolgozo
- Munkakorok
- Felettes

Az **Ugyfel** egyedben eltárolásra kerülnek az ügyfél személyes adatai: név, személyigazolvány szám, születési idő, lakcím, ami egy összetett tulajdonság (irányítószám, város, utca, házszám), valamint az ügyfelek látogatásának időpontjai is szerepelnek, belépési- és kilépési időponttal.

A **Dolgozo** egyed a dolgozók adatait tartalmazza, ezek pedig az Ugyfel egyedhez hasonlóan a munkavállaló nevét, személyigazolvány számát, születési idejét, és lakcímét (irányítószám, város, utca, házszám – ez összetett tulajdonság) jelenti.

A dolgozók különböző munkaköröket töltenek be, ezek találhatóak a **Munkakorok** egyedben. Egy alkalmazottnak több munkakörre is lehet képesítése és egy munkakört több dolgozó is betölthet, ez egy N:M kapcsolat, melynek elnevezése **Szerep**, ez a kapcsolat pedig rendelkezik egy *helyettesit-e* tulajdonsággal (0 vagy 1 értékkel).

A munkakörök megnevezése és az azokhoz szükséges ismeretek szintén részei a modellnek. Egyes munkakörökhöz több előírt ismeret is szükséges, ezeket pedig az *ismeretek* többértékű tulajdonság tartalmazza. Ez az okmányiroda xml-jében **Munkakor_ism** néven szerepel elementként.

Az okmányirodában számos ablak található, mindezek különféle ügykörök ügyintézésére szolgálnak. Az **Ablak** egyed tulajdonságai az *aid, ugykor* és a *did* idegenkulcs. Egy ablakban egyszerre egy dolgozó tartózkodhat, ez egy 1:1 kapcsolat *Beosztott* elnevezéssel. Az adott ablaknál több ügyfél is lehet egyszerre, ugyanis előfordulhat, hogy például egy házaspár intézni kívánja közös ügyleteit. Ez egy 1:N kapcsolatot jelent *Latogatas* névvel.

A **Felettes** egyed a Dolgozo egyeddel van egy-több kapcsolatban, melynek elnevezése *Fonoke*. Egy felettes a modell szerint rendelkezik névvel és személyigazolvánnyal. Egy főnök lehet több dolgozó felettese, de egy dolgozónak csak egy felettese lehet.

A modell alapján az okmányiroda XML-jének elementjei az alábbiak:

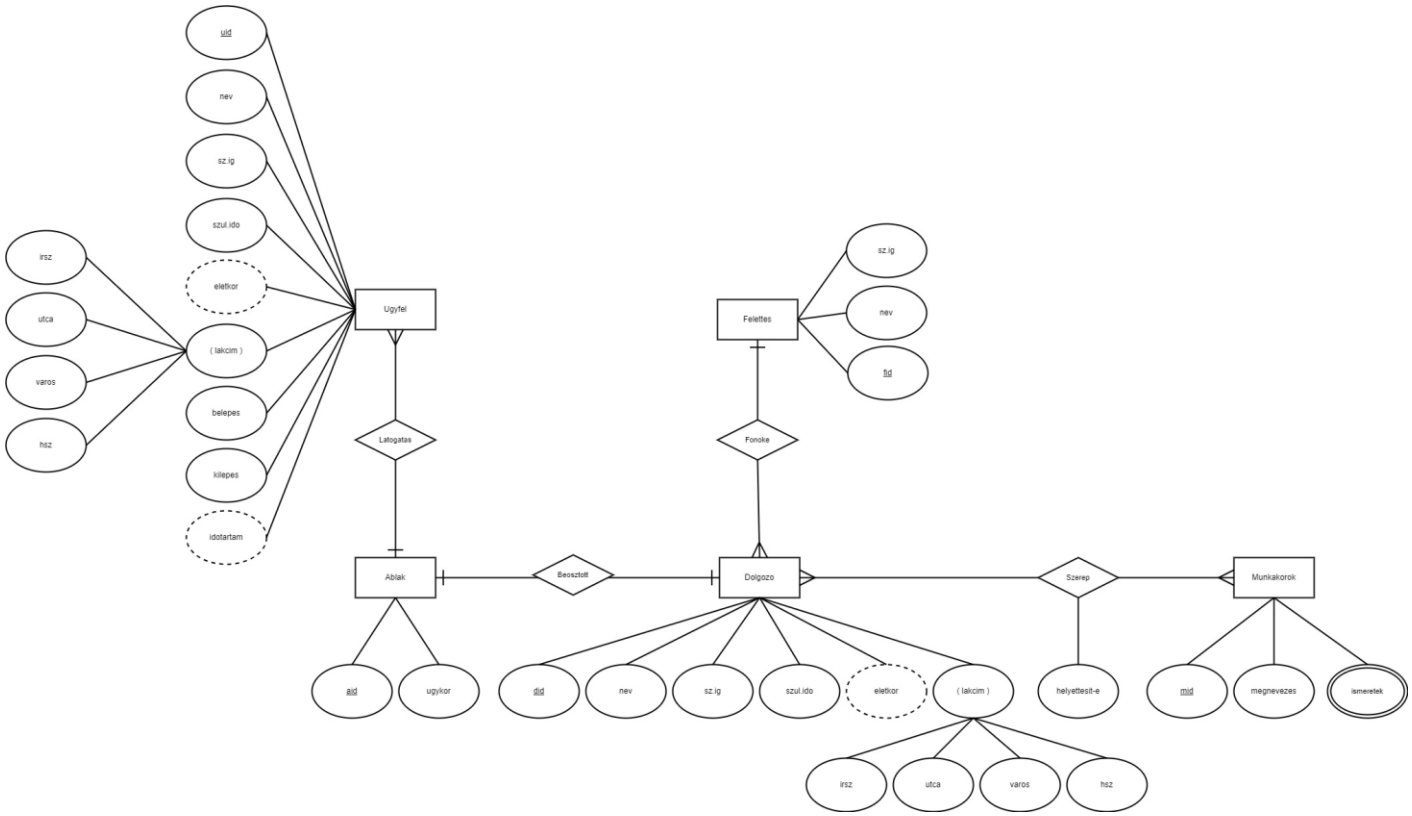
- ugyfel
- dolgozo
- ablak
- munkakorok

- munkakor_ism
- szerep
- felettes

1. feladat

1a) Az adatbázis ER modell tervezése

Az okmányiroda ER modellje az alábbi képen látható, ezt a draw.io segítségével készítettem el:



Egyedek:

1. Ugyfel egyed

   Az Ugyfel egyed tulajdonságai a következők: *uid* elsődleges kulcs, *nev, szig, szulido, eletkor* (számított tulajdonság), *lakcim* összetett tulajdonság (*irsz, varos, utca, hsz*), *belepes, kilepes, idotartam* számított tulajdonság. Tartalmazza még az Ablak egyed idegenkulcsát is, *aid* névvel.

2. Ablak egyed

   Az Ablak egyed tulajdonságai a következők: *aid* elsődleges kulcs, *ugykor*. Tartalmazza ezen kívül a Dolgozo egyed idegenkulcsát, *did* néven.

3. Dolgozo egyed
   A Dolgozo egyed tulajdonságai a következők: *did* elsődleges kulcs, *nev, szig, szulido, eletkor* számított tulajdonság, *lakcim* összetett tulajdonság (*irsz, varos, utca, hsz*). Tartalmazza ezen kívül a Felettes tábla *fid* nevezetű idegenkulcsát is. Az egyed

elsődleges kulcsa az Ablak egyednél, valamint a Szerep kapcsolótáblában idegenkulcsként szerepel.

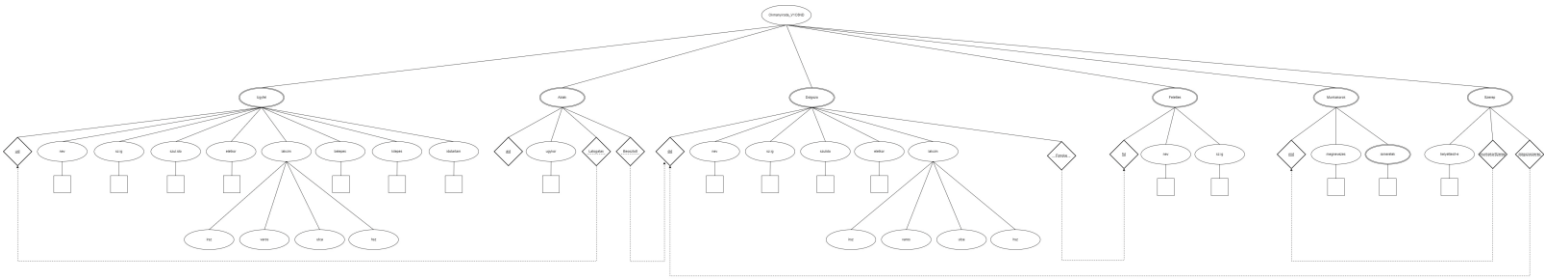
4. Felettes egyed
   A Felettes egyed tulajdonságai a következők: ***fid*** elsődleges kulcs, *nev, szig*. Az egyed elsődleges kulcsa a Dolgozo egyednél idegenkulcsként szerepel.
5. Munkakorok egyed
   A Munkakorok egyed tulajdonságai a következők: ***mid*** elsődeges kulcs, *megnevezés, ismeretek* többértékű tulajdonság. Az egyed elsődleges kulcsa a Szerep kapcsolótáblában is szerepel. Az *ismeretek* tulajdonság külön táblát alkot Munkakor_ism néven, mivel többértékű. Ez tartalmazza a Munkakorok egyed elsődleges kulcsát (***mid***), valamint a *megnevezes* tulajdonságot. Egy munkakörhöz több munkakör ismeret is tartozhat, valamint egy ismeret többféle munkakörnél is előfordulhat.

Kapcsolatok:

1. Beosztott kapcsolat
   A Beosztott kapcsolat egy 1:1 kapcsolat, amely az Ablak és Dolgozo táblát köti össze. Nem rendelkezik saját tulajdonsággal.
2. Latogatas kapcsolat
   A Latogatas kapcsolat egy 1:N kapcsolat, ami az Ablak és Ugyfel egyedeket köti össze. Nem rendelkezik saját tulajdonsággal.
3. Fonoke kapcsolat
   A Fonoke kapcsolat egy 1:N kapcsolat, amely a Dolgozo és a Felettes egyedeket köti össze. Nem rendelkezik saját tulajdonsággal.
4. Szerep kapcsolat
   A Szerep kapcsolat egy N:M kapcsolat, amely a Dolgozo és Munkakorok egyedeket köti össze. Saját tulajdonsága a *helyettesit-e*, melynek értéke 0 vagy 1 lehet, azaz igaz vagy hamis. A kapcsolat egy kapcsolótáblát hoz létre, melynek két idegenkulcsa a Dolgozo és a Munkakorok egyed elsődleges kulcsai, pontosabban a ***did*** és ***mid*** kulcsok.

1b) Az adatbázis konvertálása XDM modellre (Csak szerkesztő programmal rajzolt ábra megfelelő, szabványos szimbólumok használata)

Az okmányiroda XDM modellje az alábbi képen látható, megvalósításához a draw.io-t vettem segítségül:



Az ER modellről XDM modellre való átalakítás a következő átalakítási sémára alapul:

- egyed → elem
- elemi tulajdonság → szöveg elem
- kulcs tulajdonság → elemjellemző + kulcs megkötés
- összetett tulajdonság → elemeket tartalmazó gyerekelem
- többértékű tulajdonság → gyerekelem ismétlődéssel
- kapcsoló tulajdonság → elemjellemző + idegen kulcs
- 1:N kapcsolat → elemjellemző + kulcs + idegen kulcs
- N:M kapcsolat → kapcsoló elem és idegen kulcsok mindkét oldalon

Az XDM modell az ER modellnél már mélyebb szinten szemlélteti az egyedek közötti kapcsolatokat. Megjelennek az elsődleges kulcsok és az valamint idegenkulcsok is. Az idegenkulcsok rámutatnak egy másik egyed elsődleges kulcsára, ezt nevezzük hivatkozásnak.

1c) Az XDM modell alapján XML dokumentum készítése:

Az előzőleg említett XDM modell alapján elkészítésre került az okmányiroda XML dokumentuma. A gyökérelem *V1C6ND_okmanyiroda*. Minden gyermek elemből létrehoásra került legalább 3 példány, egyes helyeken a későbbi feladatok megoldásának megkönnyítése érdekében több is. Ezen elemek attribútumai a kulcsok és esetlegesen az idegenkulcsok.

Az XML file kódja a következő:

```
<?xml version="1.0" encoding="UTF-8"?>
<V1C6ND_okmanyiroda xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="XMLSchemaV1C6ND.xsd">

    <!-- DOLGOZÓK -->

    <dolgozo did="1" fid="1">
        <szig>14725AE</szig>
        <nev>Tóth Erzsébet</nev>
        <szulido>1985-07-11</szulido>
        <irsz>3300</irsz>
        <varos>Eger</varos>
        <utca>Széchenyi út</utca>
```

```
        <hsz>2</hsz>
    </dolgozo>

    <dolgozo did="2" fid="2">
        <szig>65815FV</szig>
        <nev>Nagy Zsigmond</nev>
        <szulido>1970-01-05</szulido>
        <irsz>3500</irsz>
        <varos>Miskolc</varos>
        <utca>Kossuth utca</utca>
        <hsz>36A</hsz>
    </dolgozo>

    <dolgozo did="3" fid="2">
        <szig>148575UK</szig>
        <nev>Kovács Levente</nev>
        <szulido>1995-12-08</szulido>
        <irsz>3434</irsz>
        <varos>Mályi</varos>
        <utca>Petőfi út</utca>
        <hsz>50</hsz>
    </dolgozo>

    <!-- további dolgozo elemek -->

    <!-- MUNKAKÖRÖK -->

    <munkakorok mid="1">
        <megnevezes>NAV ügyintéző</megnevezes>
    </munkakorok>

    <munkakorok mid="2">
        <megnevezes>gépjármű ügyintéző</megnevezes>
    </munkakorok>

    <munkakorok mid="3">
        <megnevezes>CSOK ügyintéző</megnevezes>
    </munkakorok>

    <munkakorok mid="4">
        <megnevezes>pályázati ügyintéző</megnevezes>
    </munkakorok>


    <!-- további munkakorok elemek -->

    <!-- SZEREP -->

    <szerep did="1" mid="2">
        <helyettesit-e>0</helyettesit-e>
    </szerep>
```

```xml
<szerep did="1" mid="3">
    <helyettesit-e>0</helyettesit-e>
</szerep>

<szerep did="1" mid="4">
    <helyettesit-e>0</helyettesit-e>
</szerep>

<szerep did="2" mid="1">
    <helyettesit-e>1</helyettesit-e>
</szerep>

<szerep did="3" mid="4">
    <helyettesit-e>0</helyettesit-e>
</szerep>

<szerep did="3" mid="2">
    <helyettesit-e>0</helyettesit-e>
</szerep>

<!-- további szerep elemek -->

<!-- MUNKAKÖR ISMERETEK -->

<munkakor_ism mid="1">
    <ismeretek>adóbevallás kezelése</ismeretek>
    <ismeretek>adózási ismeretek</ismeretek>
</munkakor_ism>

<munkakor_ism mid="2">
    <ismeretek>MS Office ismeretek</ismeretek>
    <ismeretek>forgalmi engedély kezelése</ismeretek>
    <ismeretek>adásvételi szerződés kezelése</ismeretek>
    <ismeretek>törzskönyv kezelése</ismeretek>
</munkakor_ism>

<munkakor_ism mid="3">
    <ismeretek>MS Office ismeretek</ismeretek>
    <ismeretek>jogi ismeretek</ismeretek>
    <ismeretek>számviteli ismeretek</ismeretek>
</munkakor_ism>

<munkakor_ism mid="4">
    <ismeretek>MS Office ismeretek</ismeretek>
    <ismeretek>jogi ismeretek</ismeretek>
</munkakor_ism>

<!-- további munkakor_ism elemek -->

<!-- ABLAK -->
```

```xml
<ablak aid="1" did="2">
    <ugykor>adóbevallás</ugykor>
</ablak>

<ablak aid="2" did="4">
    <ugykor>személyig.</ugykor>
</ablak>

<ablak aid="3" did="3">
    <ugykor>személyes iratok</ugykor>
</ablak>

<!-- további ablak elemek -->

<!-- ÜGYFÉL -->

<ugyfel uid="1" aid="3">
    <szig>577403KE</szig>
    <nev>Hegyi Károly</nev>
    <szulido>1969-09-15</szulido>
    <irsz>3432</irsz>
    <varos>Emőd</varos>
    <utca>Kossuth út</utca>
    <hsz>15</hsz>
    <belepes>2022-11-18T08:06:00</belepes>
    <kilepes>2022-11-18T08:24:00</kilepes>
</ugyfel>

<ugyfel uid="2" aid="1">
    <szig>775491HB</szig>
    <nev>Kis Elek</nev>
    <szulido>1979-04-05</szulido>
    <irsz>3300</irsz>
    <varos>Eger</varos>
    <utca>Petőfi út</utca>
    <hsz>6B</hsz>
    <belepes>2022-11-18T08:10:00</belepes>
    <kilepes>2022-11-18T08:25:00</kilepes>
</ugyfel>

<ugyfel uid="3" aid="2">
    <szig>224546DF</szig>
    <nev>Tóth Mária</nev>
    <szulido>1980-12-25</szulido>
    <irsz>3400</irsz>
    <varos>Mezőkövesd</varos>
    <utca>Mátyás király út</utca>
    <hsz>189</hsz>
    <belepes>2022-11-18T08:11:00</belepes>
    <kilepes>2022-11-18T08:29:00</kilepes>
```

```
    </ugyfel>

    <!-- további ugyfel elemek -->

    <!-- FELETTES -->
    <felettes fid="1">
        <szig>468534WW</szig>
        <nev>Piros Rózsa</nev>
    </felettes>

    <felettes fid="2">
        <szig>795341AC</szig>
        <nev>Szabó Tamás</nev>
    </felettes>

    <felettes fid="3">
        <szig>861888JB</szig>
        <nev>Lengyel László</nev>
    </felettes>

    <!-- további felettes elemek -->

</V1C6ND_okmanyiroda>
```

1d) Az XML dokumentum alapján XMLSchema készítése - saját típusok, ref, key, keyref, speciális elemek.

Egyes megkötések szükségesek ahhoz, hogy létrejöjjön egy teljes, valid struktúra, ezért elengedhetetlen egy XMLSchema-t is készíteni az XML file mellé, ami leírja az XML dokumentum felépítéséhez szükséges elemeket, valamint az elemek típusait. Mindemellett lekorlátozza bizonyos elemek számosságát is, ahol szükséges. Definiálásra kerülnek benne az elsődleges, valamint idegenkulcsok is, amik összeköttetést biztosítanak az elemek között. Ezek mind segítenek a kapcsolatok fenntartásában, és biztosítják az adatok integritását.

Az XMLSchema kódja a következő:

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

    <xs:element name="szig" type="xs:string"/>
    <xs:element name="nev" type="xs:string"/>
    <xs:element name="szulido" type="xs:date"/>
    <xs:element name="irsz" type="xs:integer"/>
    <xs:element name="varos" type="xs:string"/>
    <xs:element name="utca" type="xs:string"/>
    <xs:element name="hsz" type="xs:string"/>
    <xs:element name="fid" type="xs:integer"/>
    <xs:element name="megnevezes" type="xs:string"/>
    <xs:element name="helyettesit-e" type="xs:integer"/>
```

```
    <xs:element name="munkakor_ism" type="munkakor_ismTípus"/>
    <xs:element name="ugykor" type="xs:string"/>
    <xs:element name="did" type="xs:integer"/>
    <xs:element name="aid" type="xs:integer"/>
    <xs:element name="belepes" type="xs:dateTime"/>
    <xs:element name="kilepes" type="xs:dateTime"/>
    <xs:element name="uid" type="xs:integer"/>
    <xs:element name="ugyfel" type="ugyfelTípus"/>
    <xs:element name="felettes" type="felettesTípus"/>
    <xs:element name="dolgozo" type="dolgozoTípus"/>
    <xs:element name="munkakorok" type="munkakorokTípus"/>
    <xs:element name="szerep" type="szerepTípus"/>
    <xs:element name="ablak" type="ablakTípus"/>

    <xs:simpleType name="helyettesit-eTípus">
        <xs:restriction base="xs:integer">
            <xs:enumeration value="0"/>
            <xs:enumeration value="1"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- Összetett típusok: -->

    <xs:complexType name="ugyfelTípus">
        <xs:sequence>
            <xs:element name="szig" type="xs:string"/>
            <xs:element name="nev" type="xs:string"/>
            <xs:element name="szulido" type="xs:string"/>
            <xs:element name="irsz" type="xs:string"/>
            <xs:element name="varos" type="xs:string"/>
            <xs:element name="utca" type="xs:string"/>
            <xs:element name="hsz" type="xs:string"/>
            <xs:element ref="belepes"/>
            <xs:element ref="kilepes"/>
        </xs:sequence>
        <xs:attribute name="uid" type="xs:integer"/>
        <xs:attribute name="aid" type="xs:integer"/>
    </xs:complexType>

    <xs:complexType name="felettesTípus">
        <xs:sequence>
            <xs:element name="szig" type="xs:string"/>
            <xs:element name="nev" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="fid" type="xs:integer"/>
    </xs:complexType>

    <xs:complexType name="dolgozoTípus">
        <xs:sequence>
            <xs:element name="szig" type="xs:string"/>
            <xs:element name="nev" type="xs:string"/>
```

```
            <xs:element name="szulido" type="xs:string"/>
            <xs:element name="irsz" type="xs:string"/>
            <xs:element name="varos" type="xs:string"/>
            <xs:element name="utca" type="xs:string"/>
            <xs:element name="hsz" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="did" type="xs:integer"/>
        <xs:attribute name="fid" type="xs:integer"/>
    </xs:complexType>

    <xs:complexType name="munkakorokTípus">
        <xs:sequence>
            <xs:element name="megnevezes"  type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="mid" type="xs:integer"/>
    </xs:complexType>

    <xs:complexType name="szerepTípus">
        <xs:sequence>
            <xs:element name="helyettesit-e" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="did" type="xs:integer"/>
        <xs:attribute name="mid" type="xs:integer"/>
    </xs:complexType>

    <xs:complexType name="munkakor_ismTípus">
        <xs:sequence>
            <xs:element name="ismeretek" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="mid" type="xs:integer"/>
    </xs:complexType>

    <xs:complexType name="ablakTípus">
        <xs:sequence>
            <xs:element name="ugykor" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="aid" type="xs:integer"/>
        <xs:attribute name="did" type="xs:integer"/>
    </xs:complexType>

    <!-- Gyökérelem: -->
    <xs:element name="V1C6ND_okmanyiroda">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="dolgozo" type="dolgozoTípus"
maxOccurs="unbounded"/>
                <xs:element name="munkakorok" type="munkakorokTípus"
maxOccurs="unbounded"/>
                <xs:element name="szerep" type="szerepTípus"
maxOccurs="unbounded"/>
```

```
                <xs:element name="munkakor_ism" type="munkakor_ismTípus"
maxOccurs="unbounded"/>
                <xs:element name="ablak" type="ablakTípus"
maxOccurs="unbounded"/>
                <xs:element name="ugyfel" type="ugyfelTípus"
maxOccurs="unbounded"/>
                <xs:element name="felettes" type="felettesTípus"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>

        <!-- Saját kulcsok, idegen kulcsok: -->

        <xs:key name="dolgozoPK">
            <xs:selector xpath="dolgozo"/>
            <xs:field xpath="@did"/>
        </xs:key>

        <xs:key name="munkakorokPK">
            <xs:selector xpath="munkakorok"/>
            <xs:field xpath="@mid"/>
        </xs:key>

        <xs:key name="szerepPK">
            <xs:selector xpath="szerep"/>
            <xs:field xpath="@did"/>
            <xs:field xpath="@mid"/>
        </xs:key>

        <xs:key name="munkakor_ismPK">
            <xs:selector xpath="munkakor_ism"/>
            <xs:field xpath="@mid"/>
        </xs:key>

        <xs:key name="ablakPK">
            <xs:selector xpath="ablak"/>
            <xs:field xpath="@aid"/>
        </xs:key>

        <xs:key name="felettesPK">
            <xs:selector xpath="felettes"/>
            <xs:field xpath="@fid"/>
        </xs:key>

        <!-- Egy-több kapcsolatok: -->
        <xs:keyref name="ugyfelFK" refer="ablakPK">
            <xs:selector xpath="ugyfel"/>
            <xs:field xpath="@aid"/>
        </xs:keyref>

        <xs:keyref name="dolgozoFK" refer="felettesPK">
```

```
            <xs:selector xpath="dolgozo"/>
            <xs:field xpath="@fid"/>
        </xs:keyref>

        <!-- Egy-egy kapcsolat: -->
        <xs:unique name="dolgozoAblak" >
            <xs:selector xpath="ablak"/>
            <xs:field xpath="@did"/>
        </xs:unique>

        <!-- Több-több kapcsolat: -->
        <xs:keyref name="dolgozoSzerep" refer="dolgozoPK">
            <xs:selector xpath="szerep"/>
            <xs:field xpath="@did"/>
        </xs:keyref>

        <xs:keyref name="munkakorokSzerep" refer="munkakorokPK">
            <xs:selector xpath="szerep"/>
            <xs:field xpath="@mid"/>
        </xs:keyref>
    </xs:element>

</xs:schema>
```

2. feladat

A feladat egy DOM program készítése az XML dokumentum adatainak adminisztrálása alapján.

2a) Adatolvasás

Fájlnév: *DOMReadV1C6ND.java*

readDolgozok metódus

Ez a metódus a "dolgozo" XML elemek beolvasását végzi el az adott XML dokumentumból. Az egyes dolgozók adatait kinyeri, majd struktúrált módon kiírja a konzolra. Paraméterei: Document document - az XML dokumentum, amelyből az adatokat beolvassa.

readMunkakorok metódus

Ez a metódus a "munkakorok" XML elemek beolvasását végzi el az adott XML dokumentumból. Az egyes munkakörök adatait kinyeri, majd struktúrált módon kiírja a konzolra. Paraméterei: Document document - az XML dokumentum, amelyből az adatokat beolvassa.

readSzerepek metódus

Ez a metódus a "szerep" XML elemek beolvasását végzi el az adott XML dokumentumból. A szerepek adatait kinyeri, majd struktúrált módon kiírja a konzolra. Paraméterei: Document document - az XML dokumentum, amelyből az adatokat beolvassa.

readMunkakorIsm metódus

Ez a metódus a "munkakor_ism" XML elemek beolvasását végzi el az adott XML dokumentumból. A munkakör ismeretek adatait kinyeri, majd struktúrált módon kiírja a konzolra. Paraméterei: Document document - az XML dokumentum, amelyből az adatokat beolvassa.

readAblakok metódus

Ez a metódus az "ablak" XML elemek beolvasását végzi el az adott XML dokumentumból. Az ablakok adatait kinyeri, majd struktúrált módon kiírja a konzolra. Paraméterei: Document document - az XML dokumentum, amelyből az adatokat beolvassa.

readUgyfelek metódus:

Ez a metódus az "ugyfel" XML elemek beolvasását végzi el az adott XML dokumentumból. Az ügyfelek adatait kinyeri, majd struktúrált módon kiírja a konzolra. Paraméterei: Document document - az XML dokumentum, amelyből az adatokat beolvassa.

readFelettesek metódus:

Ez a metódus a "felettes" XML elemek beolvasását végzi el az adott XML dokumentumból. A felettesek adatait kinyeri, majd struktúrált módon kiírja a konzolra. Paraméterei: Document document - az XML dokumentum, amelyből az adatokat beolvassa.

printElement metódus:

Ez a segédmetódus egy XML elem kiíratásáért felelős a konzolra. Az elem nevét és értékét kapja paraméterként. Paraméterei: String name - az elem neve, String value - az elem értéke.

A *DOMReadV1C6ND.java* kódja a következő:

```
package XMLTaskV1C6ND.DOMParseV1C6ND.src.hu.domparse;

import org.xml.sax.SAXException;
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;

public class DOMReadV1C6ND {
    public static void main(String[] args) throws
ParserConfigurationException, SAXException, IOException {
```

```
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder = factory.newDocumentBuilder();

        // XML fájl beolvasása és dokumentummá alakítása:
        Document inputDocument = documentBuilder.parse(new
File("XMLTaskV1C6ND\\\\XMLV1C6ND.xml"));

        // A bemeneti fájl normalizálása:
        inputDocument.getDocumentElement().normalize();

        // Prolog kiíratása:
        System.out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");

        // Nyitó tag kiíratása:
        System.out.println(
                "<V1C6ND_okmanyiroda
xmlns:xs=\"http://www.w3.org/2001/XMLSchema-instance\"
xs:noNamespaceSchemaLocation=\"XMLSchemaV1C6ND.xsd\">");

        // Elementek beolvasása sorban:
        readDolgozok(inputDocument);
        readMunkakorok(inputDocument);
        readSzerepek(inputDocument);
        readMunkakorIsm(inputDocument);
        readAblakok(inputDocument);
        readUgyfelek(inputDocument);
        readFelettesek(inputDocument);

        // Záró tag kiíratása:
        System.out.println("\n</V1C6ND_okmanyiroda>");
    }

    // XML dokumentum elementjeinek beolvasása, majd struktuált módon való
kiíratása
    // konzolra:

    // Dolgozók beolvasása:
    private static void readDolgozok(Document document) {
        NodeList dolgozoList = document.getElementsByTagName("dolgozo");
        for (int i = 0; i < dolgozoList.getLength(); i++) {
            Node node = dolgozoList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element el = (Element) node;
                String did = el.getAttribute("did");
                String szig =
el.getElementsByTagName("szig").item(0).getTextContent();
                String nev =
el.getElementsByTagName("nev").item(0).getTextContent();
                String szulido =
el.getElementsByTagName("szulido").item(0).getTextContent();
```

```
                String irsz =
el.getElementsByTagName("irsz").item(0).getTextContent();
                String varos =
el.getElementsByTagName("varos").item(0).getTextContent();
                String utca =
el.getElementsByTagName("utca").item(0).getTextContent();
                String hsz =
el.getElementsByTagName("hsz").item(0).getTextContent();
                String fid = el.getAttribute("fid");

                // Kiíratás konzolra:
                System.out.println("    <dolgozo did=\"" + did + "\" fid=\"" +
fid + "\" >");
                printElement("szig", szig);
                printElement("nev", nev);
                printElement("szulido", szulido);
                printElement("irsz", irsz);
                printElement("varos", varos);
                printElement("utca", utca);
                printElement("hsz", hsz);
                System.out.println("    </dolgozo>");
            }
        }
    }

    // Munkakörök beolvasása:
    private static void readMunkakorok(Document document) {
        NodeList munkakorokList = document.getElementsByTagName("munkakorok");
        for (int i = 0; i < munkakorokList.getLength(); i++) {
            Node node = munkakorokList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element el = (Element) node;
                String mid = el.getAttribute("mid");
                String megnevezes =
el.getElementsByTagName("megnevezes").item(0).getTextContent();

                // Kiíratás konzolra:
                System.out.println("    <munkakorok mid=\"" + mid + "\">");
                printElement("megnevezes", megnevezes);
                System.out.println("    </munkakorok>");
            }
        }
    }

    // Szerepek beolvasása:
    private static void readSzerepek(Document document) {
        NodeList szerepList = document.getElementsByTagName("szerep");
        for (int i = 0; i < szerepList.getLength(); i++) {
            Node node = szerepList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element el = (Element) node;
```

```
                String did = el.getAttribute("did");
                String mid = el.getAttribute("mid");
                String helyettesit = el.getElementsByTagName("helyettesit-
e").item(0).getTextContent();

                // Kiíratás konzolra:
                System.out.println("    <szerep did=\"" + did + "\" mid=\"" +
mid + "\">");
                printElement("helyettesit-e", helyettesit);
                System.out.println("    </szerep>");
            }
        }
    }

    // Munkakör ismeretek beolvasása:
    private static void readMunkakorIsm(Document document) {
        NodeList munkakorIsmList =
document.getElementsByTagName("munkakor_ism");
        for (int i = 0; i < munkakorIsmList.getLength(); i++) {
            Node node = munkakorIsmList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element el = (Element) node;
                String mid = el.getAttribute("mid");
                System.out.println("    <munkakor_ism mid=\"" + mid + "\">");

                // Többértékű tulajdonság, így ki kell írni minden példányát:
                NodeList ismeretekList = el.getElementsByTagName("ismeretek");
                for (int j = 0; j < ismeretekList.getLength(); j++) {
                    Node ismeretekNode = ismeretekList.item(j);
                    if (ismeretekNode.getNodeType() == Node.ELEMENT_NODE) {
                        String ismeret = ismeretekNode.getTextContent();
                        printElement("ismeretek", ismeret);
                    }
                }
                System.out.println("    </munkakor_ism>");
            }
        }
    }

    // Ablakok beolvasása:
    private static void readAblakok(Document document) {
        NodeList ablakList = document.getElementsByTagName("ablak");
        for (int i = 0; i < ablakList.getLength(); i++) {
            Node node = ablakList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element el = (Element) node;
                String aid = el.getAttribute("aid");
                String ugykor =
el.getElementsByTagName("ugykor").item(0).getTextContent();
                String did = el.getAttribute("did");
```

```
                // Kiíratás konzolra:
                System.out.println("    <ablak aid=\"" + aid + "\" did=\"" +
did + "\">");
                printElement("ugykor", ugykor);
                System.out.println("    </ablak>");
            }
        }
    }

    // Ügyfelek beolvasása:
    private static void readUgyfelek(Document document) {
        NodeList ugyfelList = document.getElementsByTagName("ugyfel");
        for (int i = 0; i < ugyfelList.getLength(); i++) {
            Node node = ugyfelList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element el = (Element) node;
                String uid = el.getAttribute("uid");
                String szig =
el.getElementsByTagName("szig").item(0).getTextContent();
                String nev =
el.getElementsByTagName("nev").item(0).getTextContent();
                String szulido =
el.getElementsByTagName("szulido").item(0).getTextContent();
                String irsz =
el.getElementsByTagName("irsz").item(0).getTextContent();
                String varos =
el.getElementsByTagName("varos").item(0).getTextContent();
                String utca =
el.getElementsByTagName("utca").item(0).getTextContent();
                String hsz =
el.getElementsByTagName("hsz").item(0).getTextContent();
                String aid = el.getAttribute("aid");
                String belepes =
el.getElementsByTagName("belepes").item(0).getTextContent();
                String kilepes =
el.getElementsByTagName("kilepes").item(0).getTextContent();

                // Kiíratás konzolra:
                System.out.println("    <ugyfel uid=\"" + uid + "\" aid=\"" +
aid + "\">");
                printElement("szig", szig);
                printElement("nev", nev);
                printElement("szulido", szulido);
                printElement("irsz", irsz);
                printElement("varos", varos);
                printElement("utca", utca);
                printElement("hsz", hsz);
                printElement("belepes", belepes);
                printElement("kilepes", kilepes);
                System.out.println("    </ugyfel>");
            }
```

```
        }
    }

    // Felettesek beolvasása:
    private static void readFelettesek(Document document) {
        NodeList felettesList = document.getElementsByTagName("felettes");
        for (int i = 0; i < felettesList.getLength(); i++) {
            Node node = felettesList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element el = (Element) node;
                String fid = el.getAttribute("fid");
                String szig =
el.getElementsByTagName("szig").item(0).getTextContent();
                String nev =
el.getElementsByTagName("nev").item(0).getTextContent();

                // Kiíratás konzolra:
                System.out.println("    <felettes fid=\"" + fid + "\">");
                printElement("szig", szig);
                printElement("nev", nev);
                System.out.println("    </felettes>");
            }
        }
    }

    // Elem kiíratási metódus:
    private static void printElement(String name, String value) {
        System.out.println("        <" + name + ">" + value + "</" + name +
">");
    }
}
```

2b) Adatmódosítás

Fájlnév: *DOMModifyV1C6ND.java*

A DOMModifyV1C6ND osztályban a main metódus módosítja egy XML fájl tartalmát. Az alábbiakban rövid leírás található a különböző módosításokról:

Dolgozó nevének Módosítása

A program módosítja egy dolgozó nevét a dokumentumban: "Tóth Erzsébet" nevét változtatja "Kiss Erzsébet"-re.

Munkakör megnevezésének megváltoztatása

Módosítja egy munkakör megnevezését: a "gépjármű ügyintéző" munkakört személygépjármű ügyintéző"-re változtatja.

Munkakör ismeretek módosítása

Módosítja az "MS Office ismeretek" kifejezést "MS 365 ismeretek"-re, azáltal hogy az "ismeretek" elem tartalmát frissíti.

Ügykör változtatása

Módosítja egy ablakhoz tartozó ügykör értékét: a "személyig." ügykört "személyig., lakcímkártya"-ra változtatja.

Ügyfél személyigazolványának módosítása

Módosítja egy ügyfél személyigazolványának számát. Példaként a "577403KE" személyigazolvány számot "254338KK"-ra változtatja.

Felettes nevének módosítása:

Módosítja egy felettes nevét a dokumentumban: "Piros Rózsa" nevét változtatja "Fehérné Piros Rózsa"-ra.

A program végül kiírja a módosított XML dokumentumot a konzolra.

A *DOMModifyV1C6ND.java* kódja a következő:

```
package XMLTaskV1C6ND.DOMParseV1C6ND.src.hu.domparse;

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

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

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

public class DOMModifyV1C6ND {

    public static void main(String[] args)
            throws ParserConfigurationException, SAXException, IOException,
TransformerException {

        // Bemeneti fájl megnyitása:
        File inputFile = new File("XMLTaskV1C6ND\\XMLV1C6ND.xml");
```

```
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder = factory.newDocumentBuilder();
        Document inputDocument = documentBuilder.parse(inputFile);

        // Módosítandó elementek beolvasása:
        Node dolgozo = inputDocument.getElementsByTagName("dolgozo").item(0);
        Node munkakorok =
inputDocument.getElementsByTagName("munkakorok").item(0);
        Node munkakor_ism =
inputDocument.getElementsByTagName("munkakor_ism").item(0);
        Node ablak = inputDocument.getElementsByTagName("ablak").item(0);
        Node ugyfel = inputDocument.getElementsByTagName("ugyfel").item(0);
        Node felettes =
inputDocument.getElementsByTagName("felettes").item(0);

        NodeList dolgozoList = dolgozo.getChildNodes();
        NodeList munkakorokList = munkakorok.getChildNodes();
        NodeList munkakor_ismList = munkakor_ism.getChildNodes();
        NodeList ablakList = ablak.getChildNodes();
        NodeList ugyfelList = ugyfel.getChildNodes();
        NodeList felettesList = felettes.getChildNodes();

        // Dolgozo element módosítása:
        for (int i = 0; i < dolgozoList.getLength(); i++) {
            Node node = dolgozoList.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                // Tóth Erzsébet nevének megváltoztatása, mert férjhez ment:
                if ("nev".equals(element.getNodeName())) {
                    if ("Tóth Erzsébet".equals(element.getTextContent())) {
                        element.setTextContent("Kiss Erzsébet");
                    }
                }
            }
        }

        // Munkakorok element módosítása:
        for (int i = 0; i < munkakorokList.getLength(); i++) {
            Node node = munkakorokList.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                // A gépjármű ügyintéző munkakör megnevezésének változtatása
(pontosítás):
                if ("megnevezes".equals(element.getNodeName())) {
                    if ("gépjármű ügyintéző".equals(element.getTextContent()))
{
                        element.setTextContent("személygépjármű ügyintéző");
```

```
                    }
                }
            }
        }

        // Munkakor_ism element módosítása:
        for (int i = 0; i < munkakor_ismList.getLength(); i++) {
            Node node = munkakor_ismList.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                // Az MS Office ismeretek módosítása MS 365 ismeretekre
(növekvő elvárások a
                // munkakörben):
                if ("ismeretek".equals(element.getNodeName())) {
                    if ("MS Office
ismeretek".equals(element.getTextContent())) {
                        element.setTextContent("MS 365 ismeretek");
                    }
                }
            }
        }

        // Ablak element módosítása:
        for (int i = 0; i < ablakList.getLength(); i++) {
            Node node = ablakList.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                // A személyigazolvány ügykör változtatása, mert összevonásra
került egy másik
                // ügykörrel:
                if ("ugykor".equals(element.getNodeName())) {
                    if ("személyig.".equals(element.getTextContent())) {
                        element.setTextContent("személyig., lakcímkártya");
                    }
                }
            }
        }

        // Ugyfel element módosítása:
        for (int i = 0; i < ugyfelList.getLength(); i++) {
            Node node = ugyfelList.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                // A szig. módosítása egy ügyfélnél, akinek megváltozott a
személyigazolvány
```

```
                // száma:
                if ("szig".equals(element.getNodeName())) {
                    if ("577403KE".equals(element.getTextContent())) {
                        element.setTextContent("254338KK");
                    }
                }
            }
        }

        // Felettes element módosítása:
        for (int i = 0; i < felettesList.getLength(); i++) {
            Node node = felettesList.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                // Piros Rózsa nevének megváltoztatása, mert férjhez ment:
                if ("nev".equals(element.getNodeName())) {
                    if ("Piros Rózsa".equals(element.getTextContent())) {
                        element.setTextContent("Fehérné Piros Rózsa");
                    }
                }
            }
        }

        // Módosítások kilistázása:

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

        // Kimeneti szöveg formázása:
        transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8"); //
Karakterkódolás beállítása
        transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-
amount", "5"); // Behúzás mértékének

           // beállítása
        transformer.setOutputProperty(OutputKeys.INDENT, "yes"); // Formázás
engedélyezése
        DOMSource source = new DOMSource(inputDocument);
        System.out.println("Módosított verzió: ");
        StreamResult consoleResult = new StreamResult(System.out);
        transformer.transform(source, consoleResult);
    }

}
```

2c) Adatlekérdezés

Fájlnév: *DOMQueryV1C6ND.java*

A *DOMQueryV1C6ND.java* egy XML fájl elemzését és lekérdezéseit végzi el, különböző DOM műveletek segítségével.

Ügyfelek személyes adatainak lekérése

Az XML fájlból kinyeri az "ugyfel" elemeket, majd beolvassa az ügyfelek ID-ját, szig. számát, nevét, születési idejét és városát. Ezeket az adatokat a konzolra írja ki.

Dolgozók szűrése születési dátum alapján

A bemeneti fájlból kinyeri a "dolgozo" elemeket, majd azok születési idejét ellenőrzi. Azokat a dolgozókat szűri ki, akik 1980 után születtek. A nevüket, születési idejüket és életkorukat íratja ki a konzolra.

Elemek ABC sorrendbe rendezése

A bemeneti XML fájlból kinyeri a "dolgozo", "felettes" és "ugyfel" elemeket, majd ezeket egy közös listába gyűjti. Ezt a listát ABC sorrendbe rendezza a nevük alapján, majd a rendezett listát kiírja a konzolra.

Munkakör ismeretek statisztikája

Az XML fájlból kinyeri a "munkakor_ism" elemeket, majd azokban szereplő ismereteket vizsgálja. Megszámolja, hogy melyik ismeretből hány darab szükséges a munkakör betöltéséhez, majd kiírja a konzolra az eredményeket.

Ügyfelek látogatásának időtartama

Az XML fájlból kinyeri az "ugyfel" elemeket, majd azokban található belepes és kilepes időpontok alapján kiszámolja az eltöltött időt percekben. Az ügyfelek nevét és az időtartamot kiírja a konzolra.

A *DOMQueryV1C6ND.java* kódja a következő:

```
package XMLTaskV1C6ND.DOMParseV1C6ND.src.hu.domparse;

import java.io.File;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Date;
import java.util.List;

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 DOMQueryV1C6ND {

    public static void main(String[] args)
            throws IOException, ParserConfigurationException, SAXException,
ParseException {

        try {

            // Bemeneti XML fájl beolvasása:
            File inFile = new File("XMLTaskV1C6ND\\XMLV1C6ND.xml");

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

            Document inputDocument = dBuilder.parse(inFile);
            inputDocument.getDocumentElement().normalize();

            // Gyökérelem konzolra íratása:
            System.out.println("\n");
            System.out.print("Root element: ");
            System.out.println(inputDocument.getDocumentElement().getNodeName(
));
            System.out.println("\n");

            // 1. Lekérdezés:
            // Az ügyfelek ID-jának, szig. számának, nevének, születési
idejüknek, valamint
            // városuknak kiíratása.
            NodeList nodeList1 = inputDocument.getElementsByTagName("ugyfel");
            System.out.println("1. Ügyfelek néhány személyes adatának
kilistázása:");
            System.out.println("_____
____");

            // Végig iterálunk az elemeken és adott elemek kiíratjuk a
konzolra:
            for (int i = 0; i < nodeList1.getLength(); i++) {
                Node n = nodeList1.item(i);
                System.out.println("\n/" + (i + 1) + "/");

                if (n.getNodeType() == Node.ELEMENT_NODE) {
```

```
                    Element element = (Element) n;
                    System.out.print("Ügyfél azonosító: ");
                    System.out.println(element.getAttribute("uid"));
                    NodeList ugyfelSzigList =
element.getElementsByTagName("szig");
                    NodeList ugyfelNevList =
element.getElementsByTagName("nev");
                    NodeList ugyfelSzulidoList =
element.getElementsByTagName("szulido");
                    NodeList ugyfelVarosList =
element.getElementsByTagName("varos");

                    for (int j = 0; j < ugyfelSzigList.getLength(); j++) {
                        Node n1 = ugyfelSzigList.item(j);
                        Node n2 = ugyfelNevList.item(j);
                        Node n3 = ugyfelSzulidoList.item(j);
                        Node n4 = ugyfelVarosList.item(j);

                        Element ugyfelSzig = (Element) n1;
                        System.out.print("Ügyfél szig. száma: ");
                        System.out.println(ugyfelSzig.getTextContent());

                        Element ugyfelNev = (Element) n2;
                        System.out.print("Ügyfél neve: ");
                        System.out.println(ugyfelNev.getTextContent());

                        Element ugyfelSzulido = (Element) n3;
                        System.out.print("Ügyfél születési ideje: ");
                        System.out.println(ugyfelSzulido.getTextContent());

                        Element ugyfelVaros = (Element) n4;
                        System.out.print("Ügyfél lakhelye: ");
                        System.out.println(ugyfelVaros.getTextContent());
                    }
                }
            }
            System.out.println("\n");

            // 2. Lekérdezés:
            // Dolgozók nevének, születési idejének kiíratása, akik 1980 után
születtek, és
            // koruk kiszámolása.
            NodeList nodeList2 =
inputDocument.getElementsByTagName("dolgozo");
            System.out.println("2. Dolgozók, akik 1980 után születtek:");
            System.out.println("_____");
            int szuletett1980utan = 0;

            for (int i = 0; i < nodeList2.getLength(); i++) {
                Node n = nodeList2.item(i);
                if (n.getNodeType() == Node.ELEMENT_NODE) {
```

```
                    Element element = (Element) n;
                    NodeList dolgozokNevList =
element.getElementsByTagName("nev");
                    NodeList dolgozoSzulidoList =
element.getElementsByTagName("szulido");

                    for (int j = 0; j < dolgozokNevList.getLength(); j++) {
                        Node node1 = dolgozokNevList.item(j);
                        Node node2 = dolgozoSzulidoList.item(j);

                        Element dolgozoNev = (Element) node1;
                        Element dolgozoSzulido = (Element) node2;

                        // Ellenőrizzük, hogy az illető 1980 után született-e:
                        String szulidoString =
dolgozoSzulido.getTextContent();
                        LocalDate szuletesiDatum =
LocalDate.parse(szulidoString, DateTimeFormatter.ISO_DATE);
                        int szuletesiEv = szuletesiDatum.getYear();

                        if (szuletesiEv > 1980) {
                            szuletett1980utan++;
                            System.out.println("\n/" + (szuletett1980utan) +
"/");


                            // Kiszámoljuk az életkorát:
                            int kor = LocalDate.now().getYear() - szuletesiEv;

                            System.out.println("Név: " +
dolgozoNev.getTextContent());
                            System.out.println("Születési idő: " +
szulidoString);
                            System.out.println("Kor: " + kor + " év");
                        }

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

            // 3. Lekérdezés:
            // A dolgozók, ügyfelek és felettesek neveinek ABC sorrendbe
rendezése.
            NodeList dolgozoList =
inputDocument.getElementsByTagName("dolgozo");
            NodeList felettesList =
inputDocument.getElementsByTagName("felettes");
            NodeList ugyfelList =
inputDocument.getElementsByTagName("ugyfel");
            System.out.println("3. Dolgozók, ügyfelek, felettesek ABC
sorrendben:");
```

```
            System.out.println("_____
___");

            // A kért elementek listájának létrehozása, feltöltése:
            List<Element> dolgozoElements = getNodeListAsList(dolgozoList);
            List<Element> felettesElements = getNodeListAsList(felettesList);
            List<Element> ugyfelElements = getNodeListAsList(ugyfelList);

            // Összesítés:
            List<Element> osszesitettLista = new ArrayList<>();
            osszesitettLista.addAll(dolgozoElements);
            osszesitettLista.addAll(felettesElements);
            osszesitettLista.addAll(ugyfelElements);

            // ABC sorrendbe rendezés:
            Collections.sort(osszesitettLista, new ElementComparator());

            for (int i = 0; i < osszesitettLista.size(); i++) {
                Element element = osszesitettLista.get(i);
                System.out.println("\n/" + (i + 1) + "/ ");
                System.out.println(element.getElementsByTagName("nev").item(0)
.getTextContent());
            }
            System.out.println("\n");

            // 4. lekérdezés:
            // A munkakör ismeretek elemzése, hogy melyik hány db munkakör
betöltéséhez
            // szükséges.
            NodeList munkakorIsmNodes =
inputDocument.getElementsByTagName("munkakor_ism");
            System.out.println("4. Munkakör ismeretek statisztikája:");
            System.out.println("_____");

            // Lista az egyedi ismeretek tárolásához:
            List<String> egyediIsmeretek = new ArrayList<>();

            // Lista az ismeretek előfordulásainak számolásához:
            List<Integer> ismeretSzamlaloLista = new ArrayList<>();

            // Munkakor_ism elemek feldolgozása:
            for (int i = 0; i < munkakorIsmNodes.getLength(); i++) {
                Element munkakorIsmElement = (Element)
munkakorIsmNodes.item(i);
                NodeList ismeretNodes =
munkakorIsmElement.getElementsByTagName("ismeretek");

                // Ismeretek számlálása:
                for (int j = 0; j < ismeretNodes.getLength(); j++) {
                    Element ismeretElement = (Element) ismeretNodes.item(j);
                    String ismeret = ismeretElement.getTextContent();
```

```
                    // Ha az ismeretet először találjuk meg, adjuk hozzá az
egyediIsmeretek
                    // listához:
                    if (!egyediIsmeretek.contains(ismeret)) {
                        egyediIsmeretek.add(ismeret);
                        ismeretSzamlaloLista.add(1); // A számolás kezdeti
értékét beállítjuk 1-re.
                    } else {
                        // Ha az ismeret már korábban előfordult, növeljük az
előfordulások számát:
                        int k = egyediIsmeretek.indexOf(ismeret);
                        int ismeretSzamlalo = ismeretSzamlaloLista.get(k);
                        ismeretSzamlaloLista.set(k, ismeretSzamlalo + 1);
                    }
                }
            }
            System.out.println("\n");

            // Eredmények kiíratása a konzolra:
            for (int i = 0; i < egyediIsmeretek.size(); i++) {
                System.out.println("/" + (i + 1) + "/ ");
                String ismeret = egyediIsmeretek.get(i);
                int count = ismeretSzamlaloLista.get(i);
                System.out.println(ismeret + ": " + count + " db munkakör
betöltéséhez szükséges.\n");
            }
            System.out.println("\n");

            // 5. lekérdezés:
            // Az ügyfelek látogatásának hosszának, valamint az ügyfelek
neveinek kiíratása.
            NodeList ugyfelNodes =
inputDocument.getElementsByTagName("ugyfel");
            System.out.println("5. Ügyfelek látogatásának hossza: ");
            System.out.println("_____");

            // Dátum formázó:
            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss");

            for (int i = 0; i < ugyfelNodes.getLength(); i++) {
                Element ugyfelElement = (Element) ugyfelNodes.item(i);

                // Ügyfelek neveinek lekérése:
                String ugyfelNev =
ugyfelElement.getElementsByTagName("nev").item(0).getTextContent();

                // Belepes és kilepes időpontok lekérése az XMLfájlból:
                String belepesString =
ugyfelElement.getElementsByTagName("belepes").item(0).getTextContent();
```

```
                String kilepesString =
ugyfelElement.getElementsByTagName("kilepes").item(0).getTextContent();

                // Időpontok átalakítása Date objektummá:
                Date belepesDate = dateFormat.parse(belepesString);
                Date kilepesDate = dateFormat.parse(kilepesString);

                // Okmányirodában töltött idő kiszámítása percekben:
                long eltoltottIdoMillis = kilepesDate.getTime() -
belepesDate.getTime();
                long eltoltottIdoPercek = eltoltottIdoMillis / (60 * 1000);

                // Eredmény kiíratása konzolra:
                System.out.println("\n/" + (i + 1) + "/");
                System.out.println(ugyfelNev + " látogatása során " +
eltoltottIdoPercek
                        + " percet töltött az okmányirodában.\n");
            }

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

    // Elementeket tartalmazó NodeList konvertálása List-té:
    private static List<Element> getNodeListAsList(NodeList nodeList) {
        // Üres lista létrehozása, amely tartalmazza az Elementeket:
        List<Element> elements = new ArrayList<>();
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                // Castolja az adott Node-ot Element típusra, majd hozzáadja a
listához:
                elements.add((Element) node);
            }
        }
        return elements;
    }

    // Comparator, ami abc sorrendbe helyezi az elemeket:
    private static class ElementComparator implements Comparator<Element> {
        @Override
        public int compare(Element e1, Element e2) {
            String name1 =
e1.getElementsByTagName("nev").item(0).getTextContent();
            String name2 =
e2.getElementsByTagName("nev").item(0).getTextContent();
```

```
            return name1.compareTo(name2); // compareTo --> Compares two
strings lexicographically.
        }
    }

}
```

2d) Adatírás – Az *XMLV1C6ND.xml* dokumentum tartalmát fa struktúra formában kiírja a *konzolra* és az *XMLV1C6ND1.xml* nevű fájlba.

Fájlnév: *DOMWriteV1C6ND.java*

XML fájl létrehozása

A kód egy XML fájlt hoz létre a megadott nevű fájlnévvel. Létrehoz egy gyökér elemet V1C6ND_okmanyiroda névvel az XML dokumentumban.

Dolgozók, Munkakörök, Szerepek, Munkakör Ismeretek, Ablakok, Ügyfelek, Felettesek elementek létrehozása

Különböző metódusok segítségével hoz létre XML elemeket a gyökér elem alatt, reprezentálva a dolgozókat, munkaköröket, stb. Például createDolgozo, createMunkakorok, createSzerep, createMunkakorIsm, createAblak, createUgyfel, createFelettes.
Pl.:
createDolgozo metódus részletes leírása:
Ez a metódus egy "*dolgozo*" elemet hoz létre a megadott tulajdonságokkal és azokat hozzáadja a gyökér elemhez.
Paraméterek:

- *outputDocument*: az aktuális XML dokumentum, amelyhez az elem hozzá lesz adva
- *did*: dolgozó azonosítója
- *szig*: személyi igazolvány száma
- *nev*: dolgozó neve
- *szulido*: születési dátum
- *irsz:* irányítószám
- *varos*: város
- *utca*: utca neve
- *hsz*: házszám
- *fid*: felettes azonosítója

Működés:
Létrehoz egy "*dolgozo*" elemet az outputDocument dokumentumhoz. Hozzáad attribútumokat (*did, fid*) és különböző gyermek elemeket (*szig, nev,* stb.).Visszaadja az elkészült "*dolgozo*" elemet.

XML tranzformáció és kiíratás

Az XML fát transzformálja (Transformer) és kiírja a konzolra és egy fájlba. A kimeneti szöveg formázva van, karakterkódolása UTF-8, és behúzásokkal tagolva.

A DOMWriteV1C6ND.java kódja a következő:

```
package XMLTaskV1C6ND.DOMParseV1C6ND.src.hu.domparse;

import java.io.File;
import java.io.IOException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DOMWriteV1C6ND {

    public static void main(String[] args)
            throws ParserConfigurationException, SAXException, IOException,
TransformerException {
        File outFile = new File("XMLTaskV1C6ND\\XMLV1C6ND1.xml");

        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = factory.newDocumentBuilder();
        Document outputDocument = docBuilder.newDocument();

        Element rootElement = outputDocument.createElementNS("XMLV1C6ND",
"V1C6ND_okmanyiroda");
        outputDocument.appendChild(rootElement);
        outputDocument.setXmlVersion("1.0");

        // Dolgozó elemek létrehozása és hozzáadása a gyökér elemhez:
        rootElement.appendChild(
                createDolgozo(outputDocument, "1", "14725AE", "Tóth Erzsébet",
"1985-07-11", "3300", "Eger",
                        "Széchenyi út", "2", "1"));
        rootElement.appendChild(
                createDolgozo(outputDocument, "2", "65815FV", "Nagy Zsigmond",
"1970-01-05", "3500", "Miskolc",
```

```
                        "Kossuth utca", "36A", "2"));
        rootElement.appendChild(
                createDolgozo(outputDocument, "3", "148575UK", "Kovács
Levente", "1995-12-08", "3434", "Mályi",
                        "Petőfi út", "50", "2"));

        // Munkakör elemek:
        rootElement.appendChild(createMunkakorok(outputDocument, "1", "NAV
ügyintéző"));
        rootElement.appendChild(createMunkakorok(outputDocument, "2",
"gépjármű ügyintéző"));
        rootElement.appendChild(createMunkakorok(outputDocument, "3", "CSOK
ügyintéző"));
        rootElement.appendChild(createMunkakorok(outputDocument, "3",
"pályázati ügyintéző"));

        // Szerep elemek:
        rootElement.appendChild(createSzerep(outputDocument, "1", "2", "0"));
        rootElement.appendChild(createSzerep(outputDocument, "1", "3", "0"));
        rootElement.appendChild(createSzerep(outputDocument, "1", "4", "0"));
        rootElement.appendChild(createSzerep(outputDocument, "2", "1", "1"));
        rootElement.appendChild(createSzerep(outputDocument, "3", "4", "0"));
        rootElement.appendChild(createSzerep(outputDocument, "3", "2", "0"));

        // Munkakör ismeretek elemek:
        rootElement.appendChild(createMunkakorIsm(outputDocument, "1",
"adóbevallás kezelése"));
        rootElement.appendChild(createMunkakorIsm(outputDocument, "1",
"adózási ismeretek"));
        rootElement.appendChild(createMunkakorIsm(outputDocument, "2", "MS
Office ismeretek"));
        rootElement.appendChild(createMunkakorIsm(outputDocument, "2",
"forgalmi engedély kezelése"));
        rootElement.appendChild(createMunkakorIsm(outputDocument, "2",
"adásvételi szerződés kezelése"));
        rootElement.appendChild(createMunkakorIsm(outputDocument, "2",
"törzskönyv kezelése"));
        rootElement.appendChild(createMunkakorIsm(outputDocument, "3", "MS
Office ismeretek"));
        rootElement.appendChild(createMunkakorIsm(outputDocument, "3", "jogi
ismeretek"));
        rootElement.appendChild(createMunkakorIsm(outputDocument, "3",
"számviteli ismeretek"));
        rootElement.appendChild(createMunkakorIsm(outputDocument, "4", "MS
Office ismeretek"));
        rootElement.appendChild(createMunkakorIsm(outputDocument, "4", "jogi
ismeretek"));

        // Ablak elemek:
        rootElement.appendChild(createAblak(outputDocument, "1",
"adóbevallás", "2"));
```

```
        rootElement.appendChild(createAblak(outputDocument, "2", "személyig.",
"4"));
        rootElement.appendChild(createAblak(outputDocument, "3", "személyes
iratok", "3"));

        // Ügyfél elemek:
        rootElement
                .appendChild(createUgyfel(outputDocument, "1", "577403KE",
"Hegyi Károly", "1969-09-15", "3432", "Emőd",
                        "Kossuth út", "15", "3", "2022-11-18T08:06:00", "2022-
11-18T08:24:00"));
        rootElement.appendChild(
                createUgyfel(outputDocument, "2", "775491HB", "Kis Elek",
"1979-04-05", "3300", "Eger", "Petőfi út",
                        "6B", "5", "2022-11-18T08:10:00", "2022-11-
18T08:25:00"));
        rootElement.appendChild(
                createUgyfel(outputDocument, "3", "224546DF", "Tóth Mária",
"1980-12-25", "3400", "Mezőkövesd",
                        "Mátyás király út", "189", "2", "2022-11-18T08:11:00",
"2022-11-18T08:29:00"));

        // Felettes elemek:
        rootElement.appendChild(createFelettes(outputDocument, "1",
"468534WW", "Piros Rózsa"));
        rootElement.appendChild(createFelettes(outputDocument, "2",
"795341AC", "Szabó Tamás"));
        rootElement.appendChild(createFelettes(outputDocument, "3",
"861888JB", "Lengyel László"));

        // XML transzformációhoz szükséges Transformer objektum
inicializálása:
        TransformerFactory transformerFactory =
TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();

        // Kimeneti szöveg formázása:
        transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8"); //
Karakterkódolás beállítása
        transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-
amount", "5"); // Behúzás mértékének

           // beállítása
        transformer.setOutputProperty(OutputKeys.INDENT, "yes"); // Formázás
engedélyezése

        // DOMSource létrehozása az outputDocument alapján:
        DOMSource source = new DOMSource(outputDocument);

        // Kiíratás konzolra és fájlba:
        StreamResult console = new StreamResult(System.out);
```

```
        StreamResult file = new StreamResult(outFile);
        transformer.transform(source, console);
        transformer.transform(source, file);
    }

    // "dolgozo" elem létrehozása a hozzátartozó tulajdonságokkal:
    private static Node createDolgozo(Document outputDocument, String did,
String szig, String nev, String szulido,
            String irsz, String varos, String utca, String hsz, String fid) {
        Element dolgozo = outputDocument.createElement("dolgozo");

        dolgozo.setAttribute("did", did);
        dolgozo.setAttribute("fid", fid);
        dolgozo.appendChild(createDolgozoElement(outputDocument, "szig",
szig));
        dolgozo.appendChild(createDolgozoElement(outputDocument, "nev", nev));
        dolgozo.appendChild(createDolgozoElement(outputDocument, "szulido",
szulido));
        dolgozo.appendChild(createDolgozoElement(outputDocument, "irsz",
irsz));
        dolgozo.appendChild(createDolgozoElement(outputDocument, "varos",
varos));
        dolgozo.appendChild(createDolgozoElement(outputDocument, "utca",
utca));
        dolgozo.appendChild(createDolgozoElement(outputDocument, "hsz", hsz));

        return dolgozo;
    }

    // Különböző adatokat tartalmazó gyermek elemek létrehozása a "dolgozo"
elemhez:
    private static Node createDolgozoElement(Document outputDocument, String
name, String value) {
        Element node = outputDocument.createElement(name);
        node.appendChild(outputDocument.createTextNode(value));

        return node;
    }

    // "munkakorok" elem létrehozása a hozzátartozó tulajdonságokkal:
    private static Node createMunkakorok(Document outputDocument, String mid,
String megnevezes) {
        Element munkakorok = outputDocument.createElement("munkakorok");

        munkakorok.setAttribute("mid", mid);
        munkakorok.appendChild(createMunkakorokElement(outputDocument,
"megnevezes", megnevezes));

        return munkakorok;
    }
```

```
    // Különböző adatokat tartalmazó gyermek elemek létrehozása a "munkakorok"
    // elemhez:
    private static Node createMunkakorokElement(Document outputDocument,
String name, String value) {
        Element node = outputDocument.createElement(name);
        node.appendChild(outputDocument.createTextNode(value));

        return node;
    }

    private static Node createSzerep(Document outputDocument, String did,
String mid, String helyettesit) {
        Element szerep = outputDocument.createElement("szerep");

        szerep.setAttribute("did", did);
        szerep.setAttribute("mid", mid);
        szerep.appendChild(createSzerepElement(outputDocument, "helyettesit-
e", helyettesit));

        return szerep;
    }

    private static Node createSzerepElement(Document outputDocument, String
name, String value) {
        Element node = outputDocument.createElement(name);
        node.appendChild(outputDocument.createTextNode(value));

        return node;
    }

    private static Node createMunkakorIsm(Document outputDocument, String mid,
String ismeretek) {
        // Az tulajdonság többértékűsége miatt van szükség erre a metódusra:
        Element munkakorIsm = findMunkakorIsmElement(outputDocument, mid);

        // Ellenőrizzük, hogy létezik-e már "munkakor_ism" elem a megadott
"mid"
        // azonosítóval:
        if (munkakorIsm == null) {
            munkakorIsm = outputDocument.createElement("munkakor_ism");
            munkakorIsm.setAttribute("mid", mid);

            // Az új "munkakor_ism" elem hozzáadásra kerül:
            outputDocument.getDocumentElement().appendChild(munkakorIsm);
        }

        // Létrehozunk egy új "ismeretek" elemet és beállítjuk annak
tartalmát:
        Element ismeretekElement = outputDocument.createElement("ismeretek");
        ismeretekElement.setTextContent(ismeretek);
```

```
        // Hozzáadjuk a "munkakor_ism" elemhez:
        munkakorIsm.appendChild(ismeretekElement);

        return munkakorIsm;
    }

    private static Node createAblak(Document outputDocument, String aid,
String ugykor, String did) {
        Element ablak = outputDocument.createElement("ablak");

        ablak.setAttribute("aid", aid);
        ablak.setAttribute("did", did);
        ablak.appendChild(createAblakElement(outputDocument, "ugykor",
ugykor));

        return ablak;
    }

    private static Node createAblakElement(Document outputDocument, String
name, String value) {
        Element node = outputDocument.createElement(name);
        node.appendChild(outputDocument.createTextNode(value));

        return node;
    }

    private static Node createUgyfel(Document outputDocument, String uid,
String szig, String nev, String szulido,
            String irsz, String varos, String utca, String hsz, String aid,
String belepes, String kilepes) {
        Element ugyfel = outputDocument.createElement("ugyfel");

        ugyfel.setAttribute("uid", uid);
        ugyfel.setAttribute("aid", aid);
        ugyfel.appendChild(createUgyfelElement(outputDocument, "szig", szig));
        ugyfel.appendChild(createUgyfelElement(outputDocument, "nev", nev));
        ugyfel.appendChild(createUgyfelElement(outputDocument, "szulido",
szulido));
        ugyfel.appendChild(createUgyfelElement(outputDocument, "irsz", irsz));
        ugyfel.appendChild(createUgyfelElement(outputDocument, "varos",
varos));
        ugyfel.appendChild(createUgyfelElement(outputDocument, "utca", utca));
        ugyfel.appendChild(createUgyfelElement(outputDocument, "hsz", hsz));
        ugyfel.appendChild(createUgyfelElement(outputDocument, "belepes",
belepes));
        ugyfel.appendChild(createUgyfelElement(outputDocument, "kilepes",
kilepes));

        return ugyfel;
    }
```

```
    private static Node createUgyfelElement(Document outputDocument, String
name, String value) {
        Element node = outputDocument.createElement(name);
        node.appendChild(outputDocument.createTextNode(value));

        return node;
    }

    private static Node createFelettes(Document outputDocument, String fid,
String szig, String nev) {
        Element felettes = outputDocument.createElement("felettes");

        felettes.setAttribute("fid", fid);
        felettes.appendChild(createFelettesElement(outputDocument, "szig",
szig));
        felettes.appendChild(createFelettesElement(outputDocument, "nev",
nev));

        return felettes;
    }

    private static Node createFelettesElement(Document outputDocument, String
name, String value) {
        Element node = outputDocument.createElement(name);
        node.appendChild(outputDocument.createTextNode(value));

        return node;
    }

    private static Element findMunkakorIsmElement(Document outputDocument,
String mid) {
        // Az összes "munkakor_ism" elem megkeresése a dokumentumból:
        NodeList munkakorIsmList =
outputDocument.getElementsByTagName("munkakor_ism");

        for (int i = 0; i < munkakorIsmList.getLength(); i++) {
            Element munkakorIsmElement = (Element) munkakorIsmList.item(i);

            // Ellenőrzés: az elem "mid" attribútuma megegyezik a keresett
"mid"-del?
            if (munkakorIsmElement.getAttribute("mid").equals(mid)) {
                // Ha igen, visszaadjuk az elemet:
                return munkakorIsmElement;
            }
        }
        return null;
    }
}
```

A fájlba írás eredménye

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<V1C6ND_okmanyiroda xmlns="XMLV1C6ND">
    <dolgozo did="1" fid="1">
        <szig>14725AE</szig>
        <nev>Tóth Erzsébet</nev>
        <szulido>1985-07-11</szulido>
        <irsz>3300</irsz>
        <varos>Eger</varos>
        <utca>Széchenyi út</utca>
        <hsz>2</hsz>
    </dolgozo>
    <dolgozo did="2" fid="2">
        <szig>65815FV</szig>
        <nev>Nagy Zsigmond</nev>
        <szulido>1970-01-05</szulido>
        <irsz>3500</irsz>
        <varos>Miskolc</varos>
        <utca>Kossuth utca</utca>
        <hsz>36A</hsz>
    </dolgozo>
    <dolgozo did="3" fid="2">
        <szig>148575UK</szig>
        <nev>Kovács Levente</nev>
        <szulido>1995-12-08</szulido>
        <irsz>3434</irsz>
        <varos>Mályi</varos>
        <utca>Petőfi út</utca>
        <hsz>50</hsz>
    </dolgozo>
    <munkakorok mid="1">
        <megnevezes>NAV ügyintéző</megnevezes>
    </munkakorok>
    <munkakorok mid="2">
        <megnevezes>gépjármű ügyintéző</megnevezes>
    </munkakorok>
    <munkakorok mid="3">
        <megnevezes>CSOK ügyintéző</megnevezes>
    </munkakorok>
    <munkakorok mid="3">
        <megnevezes>pályázati ügyintéző</megnevezes>
    </munkakorok>
    <szerep did="1" mid="2">
        <helyettesit-e>0</helyettesit-e>
    </szerep>
    <szerep did="1" mid="3">
        <helyettesit-e>0</helyettesit-e>
    </szerep>
    <szerep did="1" mid="4">
        <helyettesit-e>0</helyettesit-e>
    </szerep>
```

```
    <szerep did="2" mid="1">
        <helyettesit-e>1</helyettesit-e>
    </szerep>
    <szerep did="3" mid="4">
        <helyettesit-e>0</helyettesit-e>
    </szerep>
    <szerep did="3" mid="2">
        <helyettesit-e>0</helyettesit-e>
    </szerep>
    <munkakor_ism mid="1">
        <ismeretek>adóbevallás kezelése</ismeretek>
        <ismeretek>adózási ismeretek</ismeretek>
    </munkakor_ism>
    <munkakor_ism mid="2">
        <ismeretek>MS Office ismeretek</ismeretek>
        <ismeretek>forgalmi engedély kezelése</ismeretek>
        <ismeretek>adásvételi szerződés kezelése</ismeretek>
        <ismeretek>törzskönyv kezelése</ismeretek>
    </munkakor_ism>
    <munkakor_ism mid="3">
        <ismeretek>MS Office ismeretek</ismeretek>
        <ismeretek>jogi ismeretek</ismeretek>
        <ismeretek>számviteli ismeretek</ismeretek>
    </munkakor_ism>
    <munkakor_ism mid="4">
        <ismeretek>MS Office ismeretek</ismeretek>
        <ismeretek>jogi ismeretek</ismeretek>
    </munkakor_ism>
    <ablak aid="1" did="2">
        <ugykor>adóbevallás</ugykor>
    </ablak>
    <ablak aid="2" did="4">
        <ugykor>személyig.</ugykor>
    </ablak>
    <ablak aid="3" did="3">
        <ugykor>személyes iratok</ugykor>
    </ablak>
    <ugyfel aid="3" uid="1">
        <szig>577403KE</szig>
        <nev>Hegyi Károly</nev>
        <szulido>1969-09-15</szulido>
        <irsz>3432</irsz>
        <varos>Emőd</varos>
        <utca>Kossuth út</utca>
        <hsz>15</hsz>
        <belepes>2022-11-18T08:06:00</belepes>
        <kilepes>2022-11-18T08:24:00</kilepes>
    </ugyfel>
    <ugyfel aid="5" uid="2">
        <szig>775491HB</szig>
        <nev>Kis Elek</nev>
```

```
        <szulido>1979-04-05</szulido>
        <irsz>3300</irsz>
        <varos>Eger</varos>
        <utca>Petőfi út</utca>
        <hsz>6B</hsz>
        <belepes>2022-11-18T08:10:00</belepes>
        <kilepes>2022-11-18T08:25:00</kilepes>
    </ugyfel>
    <ugyfel aid="2" uid="3">
        <szig>224546DF</szig>
        <nev>Tóth Mária</nev>
        <szulido>1980-12-25</szulido>
        <irsz>3400</irsz>
        <varos>Mezőkövesd</varos>
        <utca>Mátyás király út</utca>
        <hsz>189</hsz>
        <belepes>2022-11-18T08:11:00</belepes>
        <kilepes>2022-11-18T08:29:00</kilepes>
    </ugyfel>
    <felettes fid="1">
        <szig>468534WW</szig>
        <nev>Piros Rózsa</nev>
    </felettes>
    <felettes fid="2">
        <szig>795341AC</szig>
        <nev>Szabó Tamás</nev>
    </felettes>
    <felettes fid="3">
        <szig>861888JB</szig>
        <nev>Lengyel László</nev>
    </felettes>
</V1C6ND_okmanyiroda>
```