JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Tárgyalás

Készítette: **Juhász Balázs**
Neptunkód: **ZUYISF**

Dátum: 2023.11.14

Tartalomjegyzék

Bevezetés

A feladat leírása: (Ez legyen legalább fél oldal!)

Az általam elkészített adatbázis egy bírósági tárgyalásokat nyilvántartó adatbázis adatmodelljét ábrázolja. A téma a valóságához közel áll, erre nagyon nagy szükség van a világot tekintve bárhol. Az adatbázisban öt egyed található: Vádlott, Ügy, Ügyvéd, Bíró és Jegyző. A Vádlott egyedből nyílik 6 darab tulajdonság. Ezek a tulajdonságok a Név, Személyi igazolvány, Lakcím, Születési idő, Kor és Priusz. A Vádlott egyedben összetett tulajdonság a Név, hiszen abból nyílik a Vezetéknév és Keresztnév tulajdonságok. A Vezetéknév és Keresztnév tulajdonságok szövegeket fognak tartalmazni. A Személyi igazolvány a Primary Key, hiszen ez az adat mindenkinél más. A Kor egy származtatott tulajdonság, kiszámolható a jelenlegi évszám és a Születési idő különbségével. A Születési idő évszámot tartalmaz. A Lakcím értelemszerűen a lakcímet tartalmazza. A Priusz pedig 0-t vagy 1-et attól függően, hogy van vagy nincs, igaz/hamis. A Vádlott és az Ügyvéd egyedek egy-több kapcsolatban állnak egymással a Védi kapcsolaton keresztül. Egy ügyvédhez több

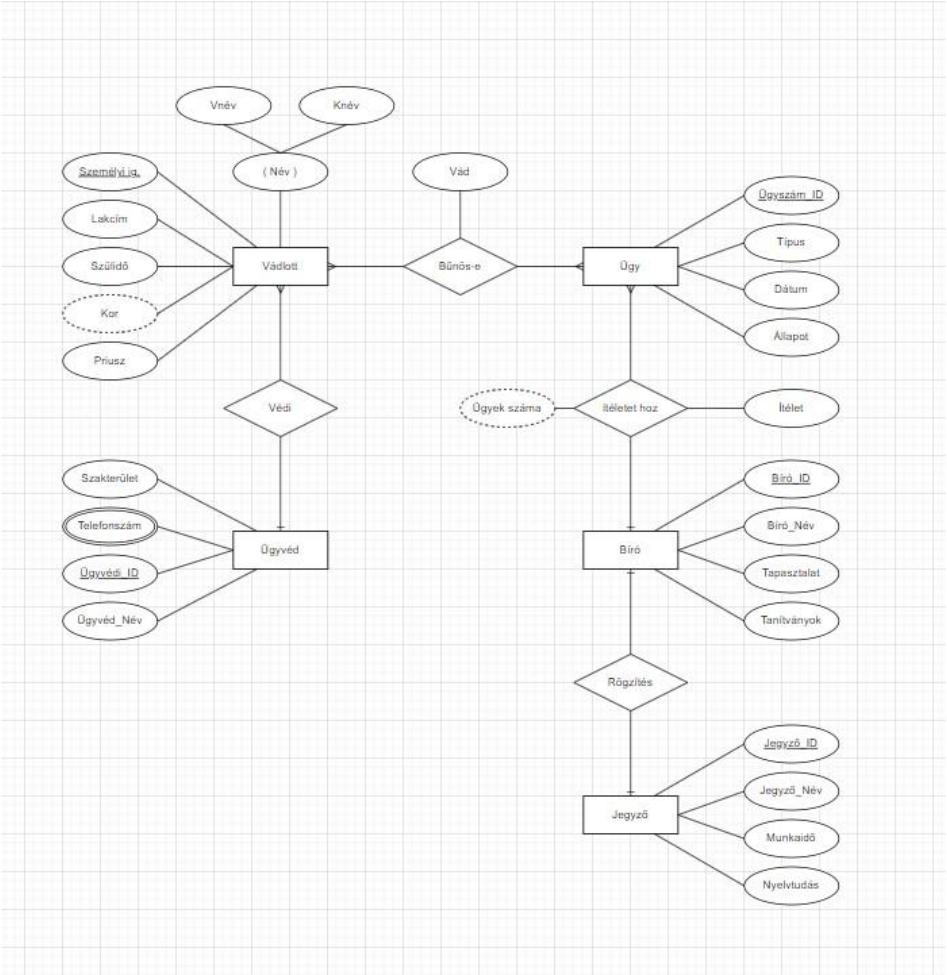
vádlott tartozhat, viszont egy vádlotthoz csak egy ügyvéd. Az Ügyvéd egyednek 4 tulajdonsága van. Ügyvédi_ID, Telefonszám, Ügyvéd_Név és Szakterület. Az Ügyvédi_ID minden ügyvédnél más és más, ezért ezzel megkülönböztethetőek, Primary Key. A Telefonszám egy többértékű tulajdonság, bárkinek lehet több telefonszáma, ez arra szolgál, hogy ezeket tartalmazza. Az Ügyvéd egyeden belül az utolsó előtti tulajdonság az Ügyvéd_név, nyilvánvalóan neveket tárol, lehetne ez is összetett, mint a fentebb említett Vádlott egyeden belüli Név, de nem így lett. Az Ügyvéd egyeden belül az utolsó tulajdonság

a Szakterület, amely az adott Ügyvéd szakterületét fogja magába foglalni szöveg típusban. A Vádlott és Ügy egyedek több-több kapcsolatban állnak egymással a Bűnös-e kapcsolaton keresztül. Egy vádlotthoz több ügy tartozhat, valamint egy ügyhöz több vádlott. Az Ügy egyednek 4 tulajdonsága van. Ügyszám_ID, Típus, Dátum és Állapot. Ügyszám_ID egy mindenhol különböző adat, amely az ügyeket számmal jelöli meg, ez a Primary Key. A Típusban lesz az Ügy konkrét típusba skatulyázása (Szabálysértési/Büntetőjogi/stb), a történés alapján. A Dátum értelemszerűen dátumokat kezel, az Ügy dátumát, hogy mikor történt az eset. Az Állapot pedig az adott Ügy állapotát tartja majd számon, folyamatban/Döntés hozva/és a többi. A Bűnös-e kapcsolaton belül van a Vád tulajdonság, amely egy szövegtípusú és értelemszerűen a vád megfogalmazását tartalmazza. Az Ügy és Bíró egyedek egy-több kapcsolatban állnak egymással az Ítéletet hoz kapcsolaton keresztül. Egy bíróhoz több ügy tartozhat, egy ügyhöz viszont egy bíró. Az ítéletet hoz kapcsolatnak két tulajdonságát emelném ki, az Ügyek számát és az Ítéletet. Az Ügyek száma egy származtatott tulajdonság, kiszámítható a bíró ügyeinek a száma. Az Ítélet tulajdonságba fog tartozni az, hogy milyen szankciót határoz meg a bíró az elítéléskor. A következő egyed a Bíró. Négy tulajdonság tartozik ide, a Bíró_ID, a Bíró_név, a Tapasztalat és a Tanítványok. A Bíró_ID egy bírónkénti eltérő szám, ezzel tudjuk azonosítani a bírót, Primary Key. A Bíró_név a bíró neve, ez is lehetne összetett is akár. A Tapasztalat tulajdonság tartalmazza, hogy hány év tapasztalattal rendelkezik az adott Bíró, például: ’5 év’. A Bíró egyeden belül az utolsó tulajdonságunk a Tanítványok, amely egy számtípus, hiszen csak a tanítványok számát fogja tárolni, akiket a Bíró tanított/képzett. A Bíró és a Jegyző egyedek egy-egy kapcsolat állnak egymással a Rögzítés kapcsolaton keresztül. Egy Bíróhoz egy Jegyző tartozik, valamint egy Jegyzőhöz egy Bíró. A Jegyző egyednek 4 tulajdonságát különböztetjük meg, ezek a Jegyző_ID, Jegyző_Név, Munkaidő, Nyelvtudás. A Jegyző_ID a Primary Key, hiszen ez mindig különböző. A Jegyző_Név logikusan a jegyző nevét foglalja magába. A Munkaidő szöveges formátum, például: Teljes munkaidő/Részleges munkaidő. A Nyelvtudás egy szöveges felsorolás, például: Angol, Francia, Német/Angol, Kínai.

1. feladat

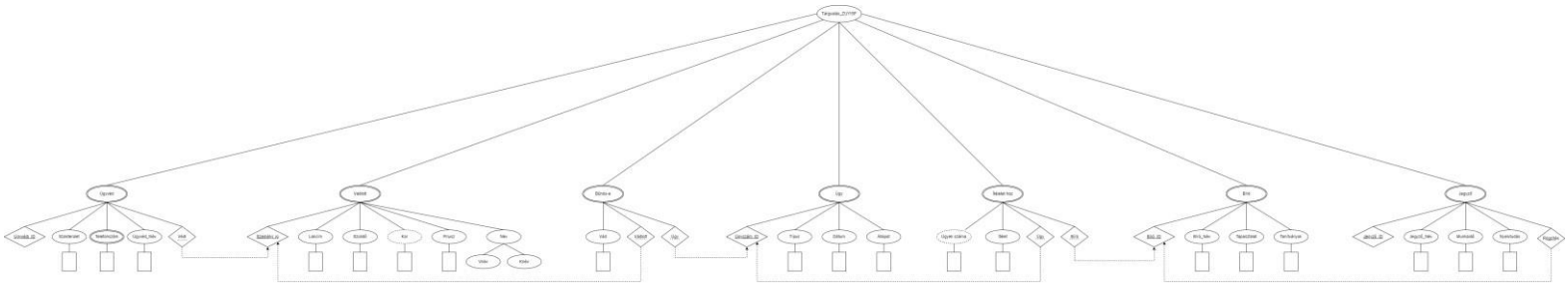
1a) Az adatbázis ER modell tervezése (Legyen legalább 5 egyed, többféle kapcsolat (1:1; 1:N; M:N), tulajdonságok - normál, kulcs, összetett, többértékű. (Csak szerkesztő programmal rajzolt ábra megfelelő, szabványos szimbólumok használata) megvalósítás rövid leírás…., majd a modell

A Tárgyalás ER modell 5 egyedből áll, mindegyik egyednek van legalább 4 tulajdonsága. Mindegyik egyed kapcsolatban van egy másik egyeddel, 1-Több, TöbbTöbb, 1-1 kapcsolatokról beszélünk ebben az esetben. Kapcsolatok közül is van, amelyiknek van saját tulajdonsága.



1b) Az adatbázis konvertálása XDM modellre (Csak szerkesztő programmal rajzolt ábra megfelelő, szabványos szimbólumok használata) megvalósítás rövid leírás…, majd a modell

Fogtam az ER modellben lévő egyedeket és tulajdonságokat, majd XDM megfelelőjére alakítottam őket, mindezt a Tárgyalás_ZUYISF séma név alatt. Kialakítottam az idegen kulcsokat és a sima kulcsokat is, majd a megfelelőket összekötöttem.



1c) Az XDM modell alapján XML dokumentum készítése**:** (Ide kerül az XML kódja!) megvalósítás rövid leírás…., majd a kód
Minden *többszörös előfordulási element*-ről legalább 3 példány készítése. A kódban megjegyzések használata.

Az XML elkészítésekor az XDM modell hasznomra volt, átlátható volt, hogy kihez milyen azonosító és tulajdonságok járnak. A többszörösen elfordulható tulajdonságok kaptak 3 példányt. A végén validáltam az XMLSchema-val együtt a https://www.liquidtechnologies.com/online-xsd-validator oldalon. Az eredmény: ’Document Valid’

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

<Tárgyalás_ZUYISF xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="XMLSchemaZUYISF.xsd">

  <!--Ügyvéd-->
  <Ügyvéd Ügyvédi_ID="1" Védi="456789AB">

    <Szakterület>Büntetőjog</Szakterület>

    <Telefonszám>+36204579450</Telefonszám>
    <Telefonszám>+36304796185</Telefonszám>
    <Telefonszám>+36507963418</Telefonszám>
```

<Ügyvéd_Név>Dr. Kovács Mihály</Ügyvéd_Név>

</Ügyvéd>

<Ügyvéd Ügyvédi_ID="2" Védi="785216CD">

<Szakterület>Szabálysértési jog</Szakterület>

<Telefonszám>+36709614250</Telefonszám>
<Telefonszám>+36704987134</Telefonszám>
<Telefonszám>+36207943157</Telefonszám>

<Ügyvéd_Név>Dr. Tóth Eszter</Ügyvéd_Név>

</Ügyvéd>

<Ügyvéd Ügyvédi_ID="3" Védi="235125AK"> <!-- Harmadik példány.-->

<Szakterület>Kiberbűnözés elleni védelem</Szakterület>

<Telefonszám>+36201234567</Telefonszám>
<Telefonszám>+36701112233</Telefonszám>
<Telefonszám>+36105556677</Telefonszám>

```
    <Ügyvéd_Név>Dr. Nagy Laura</Ügyvéd_Név>

</Ügyvéd>

<!--Vádlott-->
<Vádlott Személyi_ig="456789AB">

    <Lakcím>1111 Budapest, Jókai Mór u. 51</Lakcím>
    <Szülidő>1971-10-11</Szülidő>
    <Kor>52</Kor>
    <Priusz>0</Priusz>
    <Név>
        <Vnév>Kovács</Vnév>
        <Knév>Ferenc</Knév>
    </Név>

</Vádlott>

<Vádlott Személyi_ig="785216CD">

    <Lakcím>3525 Miskolc, Aba u. 12</Lakcím>
    <Szülidő>1999-05-21</Szülidő>
    <Kor>24</Kor>
    <Priusz>1</Priusz>
    <Név>
```

```
        <Vnév>Kiss</Vnév>
        <Knév>Tamás</Knév>
    </Név>

</Vádlott>

<Vádlott Személyi_ig="235125AK"> <!-- Harmadik példány.-->

    <Lakcím>3545 Szerencs, Napsugár u. 35.</Lakcím>
    <Szülidő>1985-11-15</Szülidő>
    <Kor>37</Kor>
    <Priusz>0</Priusz>
    <Név>
        <Vnév>Nagy</Vnév>
        <Knév>Márta</Knév>
    </Név>

</Vádlott>

<!--Bűnös-e-->
<Bűnös-e Vádlott="456789AB" Ügy="3000">

    <Vád>Súlyos testi sértés</Vád>

</Bűnös-e>
```

<Bűnös-e Vádlott="785216CD" Ügy="3001">

<Vád>Közlekedési szabálysértés</Vád>

</Bűnös-e>

<Bűnös-e Vádlott="235125AK" Ügy="3002"> <!-- Harmadik példány.-->

<Vád>Kiberbűnözés, adathalászat és rendszeres támadás</Vád>

</Bűnös-e>

<!--Ügy-->
<Ügy Ügyszám_ID="3000">

<Típus>Büntetőjogi eljárás</Típus>

<Dátum>2023-11-09</Dátum>

<Állapot>Elbírálva</Állapot>

</Ügy>

<Ügy Ügyszám_ID="3001">

<Típus>Szabálysértési eljárás</Típus>

<Dátum>2023-08-16</Dátum>

<Állapot>Elbírálás alatt</Állapot>

</Ügy>

<Ügy Ügyszám_ID="3002"> <!-- Harmadik példány.-->

<Típus>Kiberbűnözés eljárás</Típus>

<Dátum>2023-10-20</Dátum>

<Állapot>Elbírálás alatt</Állapot>

</Ügy>

<!--Ítéletet_hoz-->

<Ítéletet_hoz Ügy="3000" Bíró="1000">

<Ügyek_száma>6</Ügyek_száma>

<Ítélet>Bűnösnek találtatuk a vádlott, 2 év felfüggesztett szabadságvesztésre ítélve.</Ítélet>

</Ítéletet_hoz>

<Ítéletet_hoz Ügy="3001" Bíró="1001">

<Ügyek_száma>4</Ügyek_száma>

<Ítélet>Bírság megfizetésére kötelezve, 100 000 forint összegben.</Ítélet>

```
    </Ítéletet_hoz>

    <Ítéletet_hoz Ügy="3002" Bíró="1002"> <!-- Harmadik példány.-->

        <Ügyek_száma>5</Ügyek_száma>
        <Ítélet>Távollétben végrehajtandó börtönbüntetés, valamint anyagi kártérítési
kötelezettség.</Ítélet>

    </Ítéletet_hoz>

    <!--Bíró-->
    <Bíró Bíró_ID="1000">

        <Bíró_Név>Dr. Varga Júlia</Bíró_Név>
        <Tapasztalat>15 év</Tapasztalat>
        <Tanítványok>3</Tanítványok>

    </Bíró>

    <Bíró Bíró_ID="1001">

        <Bíró_Név>Dr. Tóth Ferenc</Bíró_Név> <!-- Második példány.-->
        <Tapasztalat>16 év</Tapasztalat>
        <Tanítványok>2</Tanítványok>
```

```
</Bíró>

<Bíró Bíró_ID="1002">

    <Bíró_Név>Dr. Kiss Emese</Bíró_Név> <!-- Harmadik példány.-->
    <Tapasztalat>20 év</Tapasztalat>
    <Tanítványok>5</Tanítványok>

</Bíró>

<!--Jegyző-->
<Jegyző Jegyző_ID="100" Rögzítés="1000">

    <Jegyző_Név>Dr. Tóth Ildikó</Jegyző_Név>
    <Munkaidő>Teljes munkaidő</Munkaidő>
    <Nyelvtudás>Angol, Német, Francia</Nyelvtudás>

</Jegyző>

<Jegyző Jegyző_ID="101" Rögzítés="1001"> <!-- Második példány.-->

    <Jegyző_Név>Dr. Magi Sándor</Jegyző_Név>
    <Munkaidő>Rész munkaidő</Munkaidő>
    <Nyelvtudás>Angol, Francia</Nyelvtudás>
```

```
    </Jegyző>

    <Jegyző Jegyző_ID="102" Rögzítés="1002"> <!-- Harmadik példány.-->

        <Jegyző_Név>Dr. Kovács János</Jegyző_Név>
        <Munkaidő>Teljes munkaidő</Munkaidő>
        <Nyelvtudás>Angol, Kínai, Francia</Nyelvtudás>

    </Jegyző>

</Tárgyalás_ZUYISF>
```

1d) Az XML dokumentum alapján XMLSchema készítése - **saját típusok, ref, key, keyref, speciális elemek.** (Ide kerül az XML Schema kódja!) megvalósítás rövid leírás…., majd a kód
A kódban *megjegyzések* használata.

Az XML dokumentum alapján az XMLSchema-t készítettem el. Használtam megjegyzéseket, ahogy az órán is tettük. Hoztam létre egyszerű/komplex típusokat. Külön megjegyzést kaptak az elsődleges kulcsok, továbbá az idegen kulcsok is. Természetesen állítottam be előfordulási értékéket (minOccurs/maxOccurs) a logikának megfelelően.

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <!--Egyszerű típusok kigyűjtése, saját típusok meghatározása, megszorítás-->
    <xs:element name="Szakterület" type="xs:string" />
    <xs:element name="Telefonszám" type="telefonszámTípus" />
    <xs:element name="Ügyvéd_Név" type="xs:string" />
```

```
<xs:element name="Lakcím" type="xs:string" />
<xs:element name="Szülidő" type="xs:date" />
<xs:element name="Kor" type="xs:int" />
<xs:element name="Priusz" type="priuszTipus" /> <!-- 1 vagy 0 | Van vagy nincs-->
<xs:element name="Vád" type="xs:string" />
<xs:element name="Típus" type="xs:string" />
<xs:element name="Dátum" type="xs:date" />
<xs:element name="Állapot" type="xs:string" />
<xs:element name="Ügyek_száma" type="xs:string" />
<xs:element name="Ítélet" type="xs:string" />
<xs:element name="Bíró_Név" type="xs:string" />
<xs:element name="Tapasztalat" type="xs:string" />
<xs:element name="Tanítványok" type="xs:int" />
<xs:element name="Jegyző_Név" type="xs:string" />
<xs:element name="Munkaidő" type="xs:string" />
<xs:element name="Nyelvtudás" type="xs:string" />


<xs:simpleType name="telefonszámTípus">
   <xs:restriction base="xs:string">
      <xs:pattern value="\+\d{11}" />
   </xs:restriction>
</xs:simpleType>


<xs:simpleType name="személyi_igTipus">
   <xs:restriction base="xs:string">
      <xs:length value="8" />
   </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="priuszTipus">
   <xs:restriction base="xs:boolean" />
</xs:simpleType>


<!--Komplex típusokhoz saját típus meghatározása, sorrendiség, számosság etc.-->
<xs:complexType name="ÜgyvédTípus">
   <xs:sequence>
      <xs:element ref="Szakterület" />
      <xs:element ref="Telefonszám" minOccurs="1" maxOccurs="unbounded" />
      <xs:element ref="Ügyvéd_Név" />
   </xs:sequence>
   <xs:attribute name="Ügyvédi_ID" type="xs:integer" use="required"/>
   <xs:attribute name="Védi" type="személyi_igTipus" use="required"/>
</xs:complexType>


<xs:complexType name="VádlottTípus">
   <xs:sequence>
      <xs:element ref="Lakcím" />
      <xs:element ref="Szülidő" />
      <xs:element ref="Kor" />
      <xs:element ref="Priusz" />
      <xs:element name="Név">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="Vnév"/>
               <xs:element name="Knév"/>
```

```
            </xs:sequence>
          </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="Személyi_ig" type="személyi_igTipus" use="required"/>
</xs:complexType>


<xs:complexType name="Bűnös-eTípus">
    <xs:sequence>
        <xs:element ref="Vád" />
    </xs:sequence>
    <xs:attribute name="Vádlott" type="személyi_igTipus" use="required"/>
    <xs:attribute name="Ügy" type="xs:integer" use="required"/>
</xs:complexType>


<xs:complexType name="ÜgyTípus">
    <xs:sequence>
        <xs:element ref="Típus" />
        <xs:element ref="Dátum" />
        <xs:element ref="Állapot" />
    </xs:sequence>
    <xs:attribute name="Ügyszám_ID" type="xs:integer" use="required" />
</xs:complexType>


<xs:complexType name="Ítéletet_hozTípus">
    <xs:sequence>
        <xs:element ref="Ügyek_száma" />
```

```
        <xs:element ref="Ítélet" />
    </xs:sequence>
    <xs:attribute name="Ügy" type="xs:integer" use="required" />
    <xs:attribute name="Bíró" type="xs:integer" use="required" />
</xs:complexType>


<xs:complexType name="BíróTípus">
    <xs:sequence>
        <xs:element ref="Bíró_Név" />
        <xs:element ref="Tapasztalat" />
        <xs:element ref="Tanítványok" />
    </xs:sequence>
    <xs:attribute name="Bíró_ID" type="xs:integer" use="required" />
</xs:complexType>


<xs:complexType name="JegyzőTípus">
    <xs:sequence>
        <xs:element ref="Jegyző_Név" />
        <xs:element ref="Munkaidő" />
        <xs:element ref="Nyelvtudás" />
    </xs:sequence>
    <xs:attribute name="Jegyző_ID" type="xs:integer" use="required" />
    <xs:attribute name="Rögzítés" type="xs:integer" use="required" />
</xs:complexType>


<!--Gyökérelemtől az elemek felhasználása-->
<xs:element name="Tárgyalás_ZUYISF">
```

```
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Ügyvéd" type="ÜgyvédTípus" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="Vádlott" type="VádlottTípus" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="Bűnös-e" type="Bűnös-eTípus" minOccurs="1"
maxOccurs="unbounded"/>
            <xs:element name="Ügy" type="ÜgyTípus" minOccurs="1"
maxOccurs="unbounded"/>
            <xs:element name="Ítéletet_hoz" type="Ítéletet_hozTípus" minOccurs="1"
maxOccurs="unbounded"/>
            <xs:element name="Bíró" type="BíróTípus" minOccurs="1"
maxOccurs="unbounded"/>
            <xs:element name="Jegyző" type="JegyzőTípus" minOccurs="1"
maxOccurs="unbounded"/>
         </xs:sequence>
      </xs:complexType>


   <!--Elsődleges kulcsok-->
   <xs:key name="Ügyvéd_kulcs">
      <xs:selector xpath="Ügyvéd" />
      <xs:field xpath="@Ügyvédi_ID" />
   </xs:key>

   <xs:key name="Vádlott_kulcs">
      <xs:selector xpath="Vádlott" />
      <xs:field xpath="@Személyi_ig" />
   </xs:key>

   <xs:key name="Ügy_kulcs">
```

```
    <xs:selector xpath="Ügy" />
    <xs:field xpath="@Ügyszám_ID" />
</xs:key>

<xs:key name="Bíró_kulcs">
    <xs:selector xpath="Bíró" />
    <xs:field xpath="@Bíró_ID" />
</xs:key>

<xs:key name="Jegyző_kulcs">
    <xs:selector xpath="Jegyző" />
    <xs:field xpath="@Jegyző_ID" />
</xs:key>


<!--Idegen kulcsok-->
<xs:keyref name="Vádlott_Ügyvéd_kulcs" refer="Vádlott_kulcs">
    <xs:selector xpath="Ügyvéd" />
    <xs:field xpath="@Védi" />
</xs:keyref>

<xs:keyref name="Bünös-e_Vádlott_kulcs" refer="Vádlott_kulcs">
    <xs:selector xpath="Bűnös-e" />
    <xs:field xpath="@Vádlott" />
</xs:keyref>

<xs:keyref name="Bünös-e_Ügy_kulcs" refer="Ügy_kulcs">
    <xs:selector xpath="Bűnös-e" />
    <xs:field xpath="@Ügy" />
</xs:keyref>
```

```
    <xs:keyref name="Ügy_Ítéletet_hoz_kulcs" refer="Ügy_kulcs">
      <xs:selector xpath="Ítéletet_hoz" />
      <xs:field xpath="@Ügy" />
    </xs:keyref>

    <xs:keyref name="Bíró_Ítéletet_hoz_kulcs" refer="Bíró_kulcs">
      <xs:selector xpath="Ítéletet_hoz" />
      <xs:field xpath="@Bíró" />
    </xs:keyref>

    <xs:keyref name="Bíró_Jegyző_kulcs" refer="Bíró_kulcs">
      <xs:selector xpath="Jegyző" />
      <xs:field xpath="@Rögzítés" />
    </xs:keyref>

    <!--Az 1:1 kapcsolat megvalósítás-->
        <xs:unique name="Jegyző_Bíró_egyegy">
      <xs:selector xpath="Bíró"/>
      <xs:field xpath="@Jegyző"/>
        </xs:unique>

    </xs:element>

</xs:schema>
```

2. feladat

A feladat egy DOM program készítése az XML dokumentum - *XMLNeptunkod.xml* – adatainak adminisztrálása alapján: (ide kerül a kód - comment együtt)

Project name: DOMParseNeptunkod

Package: hu.domparse.neptunkod

Class names: (DomReadNeptunkod, DomModifyNeptunkod, DomQueryNeptunkod, DOMWriteNeptunkod)

2a) adatolvasás (kód – comment együtt) – fájlnév: *DOMReadNeptunkod.java* megvalósítás rövid leírás…., majd a kód
A teljes dokumentum feldolgozása és kiírása strukturált formában a konzolra, ill. mentés fájlba.

A kódban megjegyzések használata.

Olvas és feldolgoz egy XML fájlt. Betölti az XMLZUYISF.xml fájlt. Létrehoz egy DocumentBuilder objektumot, mely elemzi az XML fájlt és létrehoz egy Document objektumot, amely a fájl DOM reprezentációját tartalmazza. A getElementByTagName metódus segítségével különböző elemeket választ ki és dolgozza fel. A printNodeList bejárja a kiválasztott elemeket (NodeList), kiírja az adott elem nevét, attribútumait és gyerekelemeit. Kommenteket is kezeli, a COMMENT_NUDE-al, kiírja a kommentek tartalmát, ha komment típusú csomópontot érzékel.

```
package hu.domparse.zuyisf;


import org.w3c.dom.Comment;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.InputSource;


import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;
```

```
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.File;
import java.io.StringWriter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;



import java.util.List;

public class DomReadZUYISF {

    public static void main(String[] args) {
        try {
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

            //XML fájl beolvasása.
            Document eredetiDoc = dBuilder.parse(new InputSource("XMLZUYISF.xml"));
            eredetiDoc.getDocumentElement().normalize();

            //Adatok előkészítése.
            List<Ugyved> ugyvedDataList = UgyvedXML(eredetiDoc);
            List<Vadlott> vadlottDataList = VadlottXML(eredetiDoc);
            List<Bunose> bunoseDataList = BunoseXML(eredetiDoc);
            List<Ugy> ugyDataList = UgyXML(eredetiDoc);
```

```
            List<Iteletethoz> IteletethozLista = IteletetHozXML(eredetiDoc);
            List<Biro> biroDataList = BiroXML(eredetiDoc);
            List<Jegyzo> jegyzoList = JegyzoXML(eredetiDoc);

            //Komment.
            List<String> kommentekList = getKommentek(eredetiDoc);

            //Új XML dokumentum létrehozása, ez lesz amivel kiírok.
            Document ujDoc = dBuilder.newDocument();

            //Felépíteni az XML fát a forrás dokumentum tartalmából.
            buildTree(ujDoc, ugyvedDataList, vadlottDataList, bunoseDataList, ugyDataList,
IteletethozLista, biroDataList, jegyzoList, kommentekList);

            //Menti az új dokumentumot az XMLZUYISF1.xml fájlba.
            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");

            //Kiírja a létrehozott XML fát a konzolra.
            Kiiras(ujDoc);

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

```
private static List<String> getKommentek(Document eredetiDoc) {
    List<String> kommentekList = new ArrayList<>();
    kommentgyujtes(eredetiDoc, kommentekList);
    return kommentekList;
}

private static void kommentgyujtes(Node node, List<String> kommentekList) {
    if (node.getNodeType() == Node.COMMENT_NODE) {
        kommentekList.add(((Comment) node).getTextContent());
    }

    NodeList children = node.getChildNodes();
    for (int i = 0; i < children.getLength(); i++) {
        kommentgyujtes(children.item(i), kommentekList);
    }
}

private static void Kiiras(Document doc) {
    try {
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");

        StringWriter writer = new StringWriter();
        transformer.transform(new DOMSource(doc), new StreamResult(writer));
```

```
            //Kiírja az XML dokumentumot a konzolra.
            System.out.println(writer.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void buildTree(
            Document doc,
            List<Ugyved> ugyvedDataList,
            List<Vadlott> vadlottDataList,
            List<Bunose> bunoseDataList,
            List<Ugy> ugyDataList,
            List<Iteletethoz> Iteletethoz,
            List<Biro> biroDataList,
            List<Jegyzo> jegyzoList,
                    List<String> kommentekList) {


                            //Gyökér elem (Tárgyalás_ZUYISF) hozzáadása.
                            Element rootElement = doc.createElement("Tárgyalás_ZUYISF");
                            rootElement.setAttribute("xmlns:xs",
"http://www.w3.org/2001/XMLSchema-instance");
                            rootElement.setAttribute("xs:noNamespaceSchemaLocation",
"XMLSchemaZUYISF.xsd");


                            boolean[] Comment = { false, false, false, false, false, false, false
}; //Boolean primitív típus, nem működik vele a referenciaátadás, ezért boolean tömb.
```

```
                addUgyved(doc, rootElement, ugyvedDataList, kommentekList,
Comment);

                addVadlott(doc, rootElement, vadlottDataList, kommentekList,
Comment);

                addBunose(doc, rootElement, bunoseDataList, kommentekList,
Comment);

                addUgys(doc, rootElement, ugyDataList, kommentekList,
Comment);

                addIteletethoz(doc, rootElement, Iteletethoz, kommentekList,
Comment);

                addBiro(doc, rootElement, biroDataList, kommentekList,
Comment);

                addJegyzo(doc, rootElement, jegyzoList, kommentekList,
Comment);


                doc.appendChild(rootElement);


            }


   private static void addUgyved(Document doc, Element rootElement, List<Ugyved>
ugyvedDataList, List<String> kommentekList, boolean[] Comment) {
      for (Ugyved ugyvedData : ugyvedDataList) {
         addUgyved(doc, rootElement, ugyvedData, kommentekList, Comment);
      }
   }


   private static void addUgyved(Document doc, Element parentElement, Ugyved
ugyvedData, List<String> kommentekList, boolean[] Comment) {
```

```
    if (kommentekList.stream().anyMatch(comment -> comment.contains("Ügyvéd")) &&
!Comment[0]) {
        Comment commentNode = doc.createComment("Ügyvéd");
        parentElement.appendChild(commentNode);
        Comment[0] = true;
    }

      Element ugyvedElement = doc.createElement("Ügyvéd");
    ugyvedElement.setAttribute("Ügyvédi_ID", ugyvedData.getUgyvediId());
    ugyvedElement.setAttribute("Védi", ugyvedData.getVedi());

    Element szakteruletElement = doc.createElement("Szakterület");
    szakteruletElement.setTextContent(ugyvedData.getSzakterulet());
    ugyvedElement.appendChild(szakteruletElement);

    for (String telefonszam : ugyvedData.getTelefonszam()) {
        Element telefonszamElement = doc.createElement("Telefonszám");
        telefonszamElement.setTextContent(telefonszam);
        ugyvedElement.appendChild(telefonszamElement);
    }

    Element nevElement = doc.createElement("Ügyvéd_Név");
    nevElement.setTextContent(ugyvedData.getUgyved_nev());
    ugyvedElement.appendChild(nevElement);

    parentElement.appendChild(ugyvedElement);
  }
```

```
   private static void addVadlott(Document doc, Element rootElement, List<Vadlott>
vadlottDataList, List<String> kommentekList, boolean[] Comment) {

        for (Vadlott vadlott : vadlottDataList) {

           addVadlott(doc, rootElement, vadlott, kommentekList, Comment);

       }

   }


   private static void addVadlott(Document doc, Element parentElement, Vadlott vadlott,
List<String> kommentekList, boolean[] Comment) {


       if (kommentekList.stream().anyMatch(comment -> comment.contains("Vádlott")) &&
!Comment[1]) {

           Comment commentNode = doc.createComment("Vádlott");

           parentElement.appendChild(commentNode);

           Comment[1] = true;

       }


       Element vadlottElement = doc.createElement("Vádlott");

       vadlottElement.setAttribute("Személyi_ig", vadlott.getSzemelyi_ig());


       Element lakcimElement = doc.createElement("Lakcím");

       lakcimElement.setTextContent(vadlott.getLakcim());

       vadlottElement.appendChild(lakcimElement);


       Element szulidoElement = doc.createElement("Szülidő");

       szulidoElement.setTextContent(vadlott.getSzulido());

       vadlottElement.appendChild(szulidoElement);
```

```
        Element korElement = doc.createElement("Kor");
        korElement.setTextContent(vadlott.getKor());
        vadlottElement.appendChild(korElement);

        Element priuszElement = doc.createElement("Priusz");
        priuszElement.setTextContent(vadlott.getPriusz());
        vadlottElement.appendChild(priuszElement);

        Element nevElement = doc.createElement("Név");
        Element vnevElement = doc.createElement("Vnév");
        vnevElement.setTextContent(vadlott.getVnev());
        Element knevElement = doc.createElement("Knév");
        knevElement.setTextContent(vadlott.getKnev());
        nevElement.appendChild(vnevElement);
        nevElement.appendChild(knevElement);
        vadlottElement.appendChild(nevElement);

        parentElement.appendChild(vadlottElement);
    }

    private static void addBunose(Document doc, Element rootElement, List<Bunose>
bunoseDataList, List<String> kommentekList, boolean[] Comment) {
        for (Bunose bunose : bunoseDataList) {
            addBunose(doc, rootElement, bunose, kommentekList, Comment);
        }
    }
```

```
    private static void addBunose(Document doc, Element parentElement, Bunose bunose,
List<String> kommentekList, boolean[] Comment) {


        if (kommentekList.stream().anyMatch(comment -> comment.contains("Bűnös-e")) &&
!Comment[2]) {
            Comment commentNode = doc.createComment("Bűnös-e");
            parentElement.appendChild(commentNode);
            Comment[2] = true;
        }


        Element bunoseElement = doc.createElement("Bűnös-e");
        bunoseElement.setAttribute("Vádlott", bunose.getVadlott());
        bunoseElement.setAttribute("Ügy", bunose.getUgy());


        Element vadElement = doc.createElement("Vád");
        vadElement.setTextContent(bunose.getVad());
        bunoseElement.appendChild(vadElement);


        parentElement.appendChild(bunoseElement);
    }


    private static void addUgys(Document doc, Element rootElement, List<Ugy> ugyDataList,
List<String> kommentekList, boolean[] Comment) {
        for (Ugy ugy : ugyDataList) {
            addUgy(doc, rootElement, ugy, kommentekList, Comment);
        };
    }
```

```
    private static void addUgy(Document doc, Element parentElement, Ugy ugy, List<String>
kommentekList, boolean[] Comment) {


        if (kommentekList.stream().anyMatch(comment -> comment.contains("Ügy")) &&
!Comment[3]) {
            Comment commentNode = doc.createComment("Ügy");
            parentElement.appendChild(commentNode);
            Comment[3] = true;
        }



        Element ugyElement = doc.createElement("Ügy");
        ugyElement.setAttribute("Ügyszám_ID", ugy.getUgyszamId());


        Element tipusElement = doc.createElement("Típus");
        tipusElement.setTextContent(ugy.getTipus());
        ugyElement.appendChild(tipusElement);


        Element dateElement = doc.createElement("Dátum");
        dateElement.setTextContent(ugy.getDatum());
        ugyElement.appendChild(dateElement);


        Element allapotElement = doc.createElement("Állapot");
        allapotElement.setTextContent(ugy.getAllapot());
        ugyElement.appendChild(allapotElement);
```

```
        parentElement.appendChild(ugyElement);
    }

    private static void addIteletethoz(Document doc, Element rootElement, List<Iteletethoz>
iteletethozList, List<String> kommentekList, boolean[] Comment) {
        for (Iteletethoz iteletethoz : iteletethozList) {
            addIteletethoz(doc, rootElement, iteletethoz, kommentekList, Comment);
        }
    }

    private static void addIteletethoz(Document doc, Element parentElement, Iteletethoz
iteletethoz, List<String> kommentekList, boolean[] Comment) {

        if (kommentekList.stream().anyMatch(comment -> comment.contains("Ítéletet_hoz"))
&& !Comment[4]) {
            Comment commentNode = doc.createComment("Ítéletet_hoz");
            parentElement.appendChild(commentNode);
            Comment[4] = true;
        }

        Element iteletethozElement = doc.createElement("Ítéletet_hoz");
        iteletethozElement.setAttribute("Ügy", iteletethoz.getUgy());
        iteletethozElement.setAttribute("Bíró", iteletethoz.getBiro());

        Element ugyekszamaElement = doc.createElement("Ügyek_száma");
        ugyekszamaElement.setTextContent(String.valueOf(iteletethoz.getUgyek_szama()));
        iteletethozElement.appendChild(ugyekszamaElement);
```

```
        Element iteletElement = doc.createElement("Ítélet");
        iteletElement.setTextContent(iteletethoz.getItelet());
        iteletethozElement.appendChild(iteletElement);

        parentElement.appendChild(iteletethozElement);
    }

    private static void addBiro(Document doc, Element rootElement, List<Biro> biroDataList,
List<String> kommentekList, boolean[] Comment) {
        for (Biro biro : biroDataList) {
            addBiro(doc, rootElement, biro, kommentekList, Comment);
        }
    }

    private static void addBiro(Document doc, Element parentElement, Biro biro, List<String>
kommentekList, boolean[] Comment) {

        if (kommentekList.stream().anyMatch(comment -> comment.contains("Bíró")) &&
!Comment[5]) {
            Comment commentNode = doc.createComment("Bíró");
            parentElement.appendChild(commentNode);
            Comment[5] = true;
        }

        Element biroElement = doc.createElement("Bíró");
        biroElement.setAttribute("Bíró_ID", biro.getBiroId());

        Element nevElement = doc.createElement("Bíró_Név");
```

```
        nevElement.setTextContent(biro.getBiro_nev());
        biroElement.appendChild(nevElement);

        Element tapasztalatokElement = doc.createElement("Tapasztalat");
        tapasztalatokElement.setTextContent(biro.getTapasztalat());
        biroElement.appendChild(tapasztalatokElement);

        Element tanitvanyokElement = doc.createElement("Tanítványok");
        tanitvanyokElement.setTextContent(biro.getTanitvanyok());
        biroElement.appendChild(tanitvanyokElement);

        parentElement.appendChild(biroElement);
    }

    private static void addJegyzo(Document doc, Element rootElement, List<Jegyzo>
jegyzoList, List<String> kommentekList, boolean[] Comment) {
        for (Jegyzo jegyzo : jegyzoList) {
            addJegyzo(doc, rootElement, jegyzo, kommentekList, Comment);
        }
    }

    private static void addJegyzo(Document doc, Element parentElement, Jegyzo jegyzo,
List<String> kommentekList, boolean[] Comment) {

        if (kommentekList.stream().anyMatch(comment -> comment.contains("Jegyzo")) &&
!Comment[6]) {
            Comment commentNode = doc.createComment("Jegyzo");
            parentElement.appendChild(commentNode);
```

```
            Comment[6] = true;
        }

        Element jegyzoElement = doc.createElement("Jegyző");
        jegyzoElement.setAttribute("Jegyző_ID", jegyzo.getJegyzo_Id());
        jegyzoElement.setAttribute("Rögzítés", jegyzo.getRogzites());

        Element nevElement = doc.createElement("Jegyző_Név");
        nevElement.setTextContent(jegyzo.getJegyzo_nev());
        jegyzoElement.appendChild(nevElement);

        Element munkaidoElement = doc.createElement("Munkaidő");
        munkaidoElement.setTextContent(jegyzo.getMunkaIdo());
        jegyzoElement.appendChild(munkaidoElement);

        Element nyelvtudasElement = doc.createElement("Nyelvtudás");
        nyelvtudasElement.setTextContent(String.join(", ", jegyzo.getNyelvtudas()));
        jegyzoElement.appendChild(nyelvtudasElement);

        parentElement.appendChild(jegyzoElement);
    }

    private static List<Ugyved> UgyvedXML(Document doc) {
        List<Ugyved> UgyvedList = new ArrayList<>();
        NodeList nodeList = doc.getElementsByTagName("Ügyvéd");

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

```
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;


                String vedi = element.getAttribute("Védi");
                String ugyvediId = element.getAttribute("Ügyvédi_ID");
                String szakterulet =
element.getElementsByTagName("Szakterület").item(0).getTextContent();


                NodeList telefonszamNodes = element.getElementsByTagName("Telefonszám");
//Többértékű.
                List<String> telefonszamList = new ArrayList<>();
                for (int j = 0; j < telefonszamNodes.getLength(); j++) {
                    Node telefonszamNode = telefonszamNodes.item(j);
                    if (telefonszamNode.getNodeType() == Node.ELEMENT_NODE) {
                        telefonszamList.add(telefonszamNode.getTextContent());
                    }
                }


                String ugyved_nev =
element.getElementsByTagName("Ügyvéd_Név").item(0).getTextContent();


                //Ugyved objektum létrehozása és hozzáadása a listához.
                UgyvedList.add(new Ugyved(vedi, ugyvediId, szakterulet, telefonszamList,
ugyved_nev));
            }
        }
        return UgyvedList;
```

```
    }

    private static List<Vadlott> VadlottXML(Document doc) {
        List<Vadlott> VadlottList = new ArrayList<>();
        NodeList nodeList = doc.getElementsByTagName("Vádlott");

        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                String szemelyi_ig = element.getAttribute("Személyi_ig");
                String lakcim =
element.getElementsByTagName("Lakcím").item(0).getTextContent();
                String szulido =
element.getElementsByTagName("Szülidő").item(0).getTextContent();
                String kor = element.getElementsByTagName("Kor").item(0).getTextContent();
                String priusz =
element.getElementsByTagName("Priusz").item(0).getTextContent();
                String vnev = element.getElementsByTagName("Vnév").item(0).getTextContent();
                String knev = element.getElementsByTagName("Knév").item(0).getTextContent();

                //Vadlott objektum létrehozása és hozzáadása a listához.
                VadlottList.add(new Vadlott(szemelyi_ig, lakcim, szulido, kor, priusz, vnev,
knev));
            }
        }
        return VadlottList;
```

```
}

private static List<Bunose> BunoseXML(Document doc) {
    List<Bunose> BunoseList = new ArrayList<>();
    NodeList nodeList = doc.getElementsByTagName("Bűnös-e");

    for (int i = 0; i < nodeList.getLength(); i++) {
        Node node = nodeList.item(i);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;

            String vadlott = element.getAttribute("Vádlott");
            String ugy = element.getAttribute("Ügy");
            String vad = element.getElementsByTagName("Vád").item(0).getTextContent();

            //Bunose objektum létrehozása és hozzáadása a listához.
            BunoseList.add(new Bunose(vadlott, ugy, vad));
        }
    }
    return BunoseList;
}

private static List<Ugy> UgyXML(Document doc) {
    List<Ugy> UgyList = new ArrayList<>();
    NodeList nodeList = doc.getElementsByTagName("Ügy");

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

```
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                String ugy = element.getAttribute("Ügyszám_ID");
                String tipus = element.getElementsByTagName("Típus").item(0).getTextContent();
                String datum =
element.getElementsByTagName("Dátum").item(0).getTextContent();
                String allapot =
element.getElementsByTagName("Állapot").item(0).getTextContent();

                //Ugy objektum létrehozása és hozzáadása a listához.
                UgyList.add(new Ugy(ugy, tipus, datum, allapot));
            }
        }
        return UgyList;
    }


    private static List<Iteletethoz> IteletetHozXML(Document doc) {
        List<Iteletethoz> IteletetHozList = new ArrayList<>();
        NodeList nodeList = doc.getElementsByTagName("Ítéletet_hoz");

        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;
```

```
            String ugy = element.getAttribute("Ügy");
            String biro = element.getAttribute("Bíró");
            String ugyek_szama =
element.getElementsByTagName("Ügyek_száma").item(0).getTextContent();
            String itelet = element.getElementsByTagName("Ítélet").item(0).getTextContent();

            //Iteletethoz objektum létrehozása és hozzáadása a listához.
            IteletetHozList.add(new Iteletethoz(ugy, biro, ugyek_szama, itelet));
        }
    }
    return IteletetHozList;
  }

  private static List<Biro> BiroXML(Document doc) {
    List<Biro> BiroDataList = new ArrayList<>();
    NodeList nodeList = doc.getElementsByTagName("Bíró");

    for (int i = 0; i < nodeList.getLength(); i++) {
      Node node = nodeList.item(i);
      if (node.getNodeType() == Node.ELEMENT_NODE) {
        Element element = (Element) node;

        String biro_Id = element.getAttribute("Bíró_ID");
        String biro_nev =
element.getElementsByTagName("Bíró_Név").item(0).getTextContent();
        String tapasztalat =
element.getElementsByTagName("Tapasztalat").item(0).getTextContent();
```

```
            String tanitvanyok =
element.getElementsByTagName("Tanítványok").item(0).getTextContent();


            //Biro objektum létrehozása és hozzáadása a listához.
            BiroDataList.add(new Biro(biro_Id, biro_nev, tapasztalat, tanitvanyok));
        }
    }
    return BiroDataList;
}



private static List<Jegyzo> JegyzoXML(Document doc) {
    List<Jegyzo> jegyzoList = new ArrayList<>();
    NodeList nodeList = doc.getElementsByTagName("Jegyző");

    for (int i = 0; i < nodeList.getLength(); i++) {
        Node node = nodeList.item(i);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;


            String jegyzo_Id = element.getAttribute("Jegyző_ID");
            String rogzites = element.getAttribute("Rögzítés");
            String jegyzo_nev =
element.getElementsByTagName("Jegyző_Név").item(0).getTextContent();
            String munkaido =
element.getElementsByTagName("Munkaidő").item(0).getTextContent();
            String nyelvtudas =
element.getElementsByTagName("Nyelvtudás").item(0).getTextContent();
```

```
            //Jegyzo objektum létrehozása és hozzáadása a listához.

            jegyzoList.add(new Jegyzo(jegyzo_Id, rogzites, jegyzo_nev, munkaido,
nyelvtudas));

          }

       }

       return jegyzoList;

    }

}
```

2b) adatmódosítás (kód – comment együtt) – fájlnév: *DOMModifyNeptunkod.java*
megvalósítás rövid leírás…., majd a kód
Az XML dokumentum példányaiból legalább 5 módosítás készítése és kiírása a konzolra.

A kódban megjegyzések használata.

A felhasználó megadja az XML-ben módosítani kívánt elem nevét, azonosítóját és a módosítandó tulajdonság nevét. Megkeresi az XML-ben az adott nevű és azonosítójú elemet, majd módosítja az adott elem egy meghatározott tulajdonságát az új értékre, amit a felhasználó ad meg. A módosított XML dokumentumot visszaírja a fájlba (felülírja). Kiírja, hogy sikeres volt-e a változtatás vagy hibára futott-e.

```
package hu.domparse.zuyisf;


import java.io.File;

import java.io.IOException;

import java.util.Scanner;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;
```

```
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.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DomModifyZUYISF {



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

	Scanner scanner = new Scanner(System.in);


	System.out.println("Adja meg az elem nevét, amit módosítani szeretne!
(Ügyvéd/Vádlott/Bűnös-e/Ügy/Ítéletet_hoz/Bíró/Jegyző):");
	String elementName = scanner.nextLine();


	printIdentifiersForElement(elementName);
	String elementId = scanner.nextLine();


	printAttributes(elementName);
```

```
        String propertyName = scanner.nextLine();

        System.out.println("Mire szeretné módosítani a(z) " + propertyName + "
tulajdonságot?");
        String newValue = scanner.nextLine();

        File inputFile = new File("XMLZUYISF.xml");
        DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
        Document doc = docBuilder.parse(inputFile);
        doc.getDocumentElement().normalize();

        NodeList elements = doc.getElementsByTagName(elementName);
        Node elementToModify = null;

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

            Node node = elements.item(i);

            //Ügyvéd
            if (node.getAttributes().getNamedItem("Ügyvédi_ID") != null &&
        node.getAttributes().getNamedItem("Ügyvédi_ID").getTextContent().equals(elementI
d) ||
                (node.getAttributes().getNamedItem("Védi") != null &&
node.getAttributes().getNamedItem("Védi").getTextContent().equals(elementId)) &&
```

```
                elementName.equals("Ügyvéd")) {
                elementToModify = node;
                break;
            }

    //Vádlott
            if (node.getAttributes().getNamedItem("Személyi_ig") != null &&
        node.getAttributes().getNamedItem("Személyi_ig").getTextContent().equals(elementId) &&
                        elementName.equals("Vádlott")) {
                        elementToModify = node;
                        break;
                }

            //Bűnös-e
            if (node.getAttributes().getNamedItem("Vádlott") != null &&
        node.getAttributes().getNamedItem("Vádlott").getTextContent().equals(elementId) ||
                        (node.getAttributes().getNamedItem("Ügy") != null &&
        node.getAttributes().getNamedItem("Ügy").getTextContent().equals(elementId)) &&
                        elementName.equals("Bűnös-e")) {
                        elementToModify = node;
                        break;
            }

            //Ügy
```

```
                if (node.getAttributes().getNamedItem("Ügyszám_ID") != null &&
        node.getAttributes().getNamedItem("Ügyszám_ID").getTextContent().equals(element
Id) &&
                                elementName.equals("Ügy")) {
                                elementToModify = node;
                                break;
                }

                //Ítéletet_hoz
                if (node.getAttributes().getNamedItem("Bíró") != null &&
        node.getAttributes().getNamedItem("Bíró").getTextContent().equals(elementId) ||
                                (node.getAttributes().getNamedItem("Ügy") != null &&
node.getAttributes().getNamedItem("Ügy").getTextContent().equals(elementId)) &&
                                elementName.equals("Ítéletet_hoz")) {
                                elementToModify = node;
                                break;
                }

                //Bíró
                if (node.getAttributes().getNamedItem("Bíró_ID") != null &&
        node.getAttributes().getNamedItem("Bíró_ID").getTextContent().equals(elementId)
&&
                                elementName.equals("Bíró")) {
                                elementToModify = node;
                                break;
```

```
                }

                //Jegyző
                if (node.getAttributes().getNamedItem("Jegyző_ID") != null &&
        node.getAttributes().getNamedItem("Jegyző_ID").getTextContent().equals(elementId)
||
                                (node.getAttributes().getNamedItem("Rögzítés") != null &&
        node.getAttributes().getNamedItem("Rögzítés").getTextContent().equals(elementId))
&&
                elementName.equals("Jegyző")) {
                                elementToModify = node;
                                break;
                }

        }


        if (elementToModify != null)
        {

                NodeList childNodes = elementToModify.getChildNodes();

                for (int j = 0; j < childNodes.getLength(); j++)
                {

                        Node childNode = childNodes.item(j);
```

```
                    if (childNode.getNodeName().equals(propertyName))
                    {
                        childNode.setTextContent(newValue);
                        break;
                    }

        }


            //Írja vissza a módosított dokumentumot
            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            DOMSource source = new DOMSource(doc);
            StreamResult result = new StreamResult(new File("XMLZUYISF.xml"));
            transformer.transform(source, result);
            System.out.println(propertyName + " tulajdonság módosítva a következő
értékre: " + newValue);


        }
        else
        {
            System.out.println("Nem található " + elementName + " az ID-val: " +
elementId);
        }


        scanner.close();
```

```
}

private static void printAttributes(String elementName) //Mindig kiírja az adott elemhez tartozó attribútumokat!
{

String attributes = "";

switch (elementName)
{

        case "Ügyvéd":
            attributes = "Szakterület, Telefonszám, Ügyvéd_Név";
            break;

        case "Vádlott":
            attributes = "Lakcím, Szülidő, Kor, Priusz, Név";
            break;

        case "Bűnös-e":
            attributes = "Vád";
            break;

    case "Ügy":
        attributes = "Típus, Dátum, Állapot";
        break;
```

```
        case "Ítéletet_hoz":
                attributes = "Ügyek_száma, Ítélet";
                break;

        case "Bíró":
                attributes = "Bíró_Név, Tapasztalat, Tanítványok";
                break;

        case "Jegyző":
                attributes = "Jegyző_Név, Munkaidő, Nyelvtudás";
                break;

        default:
                attributes = "Nincs ilyen tulajdonság!";

    }

    System.out.println("Adja meg a(z) " + elementName + " valamelyik tulajdonságát (" +
attributes + "):");

}

private static void printIdentifiersForElement(String elementName) //Mindig kiírja az adott
elemhez tartozó azonosítókat!
{

String identifiers = "";
```

```
switch (elementName)

{

        case "Ügyvéd":
            identifiers = "Ügyvédi_ID, Védi";
            break;

        case "Vádlott":
            identifiers = "Személyi_ig";
            break;

        case "Bűnös-e":
            identifiers = "Vádlott, Ügy";
            break;

        case "Ügy":
            identifiers = "Ügyszám_ID";
            break;

        case "Ítéletet_hoz":
            identifiers = "Bíró, Ügy";
            break;

        case "Bíró":
```

```
                identifiers = "Bíró_ID";
                break;

            case "Jegyző":
                identifiers = "Jegyző_ID, Rögzítés";
                break;

            default:
                identifiers = "Nincs ilyen elem vagy azonosítók nem definiáltak";

}

System.out.println("Adja meg a(z) " + elementName + " valamelyik azonosítóját (" +
identifiers + "):");
}

}
```

2c) adatlekérdezés (kód – comment együtt) – fájlnév: *DOMQueryNeptunkod.java*
megvalósítás rövid leírás…. majd a kód.
Az XML dokumentum példányaiból legalább 5 lekérdezés készítése és kiírása a konzolra.

A lekérdezésnél NE használjon xPath kifejezéseket!

A kódban megjegyzések használata.

A felhasználó beírja az XML-ben lekérdezni kívánt elem nevét, megkeresi az összes olyan elemet az XML dokumentumban, amelyek megegyeznek a felhasználó által megadott névvel. Kiírja az adott elemek számát, azok attribútumait és gyerek elemeit.

```
package hu.domparse.zuyisf;
```

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;
import java.io.File;
import java.util.Scanner;

public class DomQueryZUYISF
{

public static void main(String[] args)
{

        Scanner scanner = new Scanner(System.in);
        System.out.println("Melyik elem adatait szeretné lekérdezni?");
   String input = scanner.nextLine();

            try
            {

                File f = new File("XMLZUYISF.xml");
              DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
              DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
              Document doc = dBuilder.parse(f);
              doc.getDocumentElement().normalize();
```

```
NodeList nodeList = doc.getElementsByTagName(input); //Megkeressük az inputhoz tartozókat.
System.out.println("Talált " + input + " elemek száma: " + nodeList.getLength());

for (int temp = 0; temp < nodeList.getLength(); temp++)
{

Node nNode = nodeList.item(temp);

if (nNode.getNodeType() == Node.ELEMENT_NODE)
{

   Element element = (Element) nNode;
   System.out.println("\nAktuális elem: " + element.getNodeName());


        if (element.hasAttributes()) //Attribútumok kiírása.
        {

              System.out.println("Attribútumok:");
          for (int i = 0; i < element.getAttributes().getLength(); ++i)
          {

              Node attr = element.getAttributes().item(i);
            System.out.println(" - " + attr.getNodeName() + ": " + attr.getNodeValue());
```

```
                }

            }

            NodeList children = element.getChildNodes(); //Gyerek elemek kiírása.
            System.out.println("Gyermek elemek:");

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

                Node child = children.item(i);

                if (child.getNodeType() == Node.ELEMENT_NODE)
                {

                    System.out.println(" - " + child.getNodeName() + ": " + child.getTextContent());

                }

            }
        }

    }
```

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

2d) adatírás - készítsen egy DOM API programot, amely egy *XMLNeptunkod.xml* dokumentum tartalmát fa struktúra formában kiírja a *konzolra és egy XMLNeptunkod1.xml* fájlba. (kód – comment együtt) – fájlnév: DOMWriteNeptunkod.java

Betölti az XMLZUYISF.xml fájlt, inicializálja a DOM parser-t a DocumentBuilderFatctory-t a DocumentBuilder segítségével. Rekurzív módon bejárja az XML dokumentumot és kiírja annak elem struktúráját, beleértve az elemeket és a szövegcsomópontokat. A módosított DOM struktúrát egy új fájlba írja (XMLZUYISF1.xml), bekezdésekkel és megfelelően formázva.

```
package hu.domparse.zuyisf;

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

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
```

```
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.File;
import java.io.StringWriter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;



import java.util.List;

class Ugyved {
    private String vedi;
    private String ugyvediId;
    private String szakterulet;
    private List<String> telefonszam;
    private String ugyved_nev;

    public Ugyved(String vedi, String ugyvediId, String szakterulet, List<String> telefonszam,
String ugyved_nev) {
        this.vedi = vedi;
        this.ugyvediId = ugyvediId;
        this.szakterulet = szakterulet;
        this.telefonszam = telefonszam;
        this.ugyved_nev = ugyved_nev;
    }

    public String getVedi() {
```

```
    return vedi;
}

public void setVedi(String vedi) {
    this.vedi = vedi;
}

public String getUgyvediId() {
    return ugyvediId;
}

public void setUgyvediId(String ugyvediId) {
    this.ugyvediId = ugyvediId;
}

public String getSzakterulet() {
    return szakterulet;
}

public void setSzakterulet(String szakterulet) {
    this.szakterulet = szakterulet;
}

public List<String> getTelefonszam() {
    return telefonszam;
}

public void setTelefonszam(List<String> telefonszam) {
    this.telefonszam = telefonszam;
}
```

```
    public String getUgyved_nev() {
        return ugyved_nev;
    }

    public void setUgyved_nev(String ugyved_nev) {
        this.ugyved_nev = ugyved_nev;
    }
}

class Vadlott {
    private String szemelyi_ig;
    private String lakcim;
    private String szulido;
    private String kor;
    private String priusz;
    private String vnev;
    private String knev;

    public Vadlott(String szemelyi_ig, String lakcim, String szulido, String kor, String priusz,
String vnev, String knev) {
        this.szemelyi_ig = szemelyi_ig;
        this.lakcim = lakcim;
        this.szulido = szulido;
        this.kor = kor;
        this.priusz = priusz;
        this.vnev = vnev;
        this.knev = knev;
    }

    public String getSzemelyi_ig() {
```

```
    return szemelyi_ig;
}

public void setSzemelyi_ig(String szemelyi_ig) {
    this.szemelyi_ig = szemelyi_ig;
}

public String getLakcim() {
    return lakcim;
}

public void setLakcim(String lakcim) {
    this.lakcim = lakcim;
}

public String getSzulido() {
    return szulido;
}

public void setSzulido(String szulido) {
    this.szulido = szulido;
}

public String getKor() {
    return kor;
}

public void setKor(String kor) {
    this.kor = kor;
}
```

```
    public String getPriusz() {
        return priusz;
    }

    public void setPriusz(String priusz) {
        this.priusz = priusz;
    }

    public String getVnev() {
        return vnev;
    }

    public void setVnev(String vnev) {
        this.vnev = vnev;
    }

    public String getKnev() {
        return knev;
    }

    public void setKnev(String knev) {
        this.knev = knev;
    }
}

class Bunose {
    private String vadlott;
    private String ugy;
    private String vad;
```

```
public Bunose(String vadlott, String ugy, String vad) {
    this.vadlott= vadlott;
    this.ugy = ugy;
    this.vad = vad;
}

public String getVadlott() {
    return vadlott;
}

public void setVadlott(String vadlott) {
    this.vadlott = vadlott;
}

public String getUgy() {
    return ugy;
}

public void setUgy(String ugy) {
    this.ugy = ugy;
}

public String getVad() {
    return vad;
}

public void setVad(String vad) {
    this.vad = vad;
}
```

```
}

class Ugy {
    private String UgyszamId;
    private String tipus;
    private String datum;
    private String allapot;

    public Ugy(String UgyszamId, String tipus, String datum, String allapot) {
        this.UgyszamId = UgyszamId;
        this.tipus = tipus;
        this.datum = datum;
        this.allapot = allapot;
    }

    public String getUgyszamId() {
        return UgyszamId;
    }

    public void setUgyszamId(String UgyszamId) {
        this.UgyszamId = UgyszamId;
    }

    public String getTipus() {
        return tipus;
    }

    public void setTipus(String tipus) {
        this.tipus = tipus;
    }
```

```
    public String getDatum() {
        return datum;
    }

    public void setDatum(String datum) {
        this.datum = datum;
    }

    public String getAllapot() {
        return allapot;
    }

    public void setAllapot(String allapot) {
        this.allapot = allapot;
    }
}

class Iteletethoz {
    private String Ugy;
    private String Biro;
    private String Ugyek_szama;
    private String Itelet;

    public Iteletethoz(String Ugy, String Biro, String ugyek_szama, String Itelet) {
        this.Ugy = Ugy;
        this.Biro = Biro;
        this.Ugyek_szama = ugyek_szama;
        this.Itelet = Itelet;
    }
```

```
public String getUgy() {
    return Ugy;
}

public void setUgy(String Ugy) {
    this.Ugy = Ugy;
}

public String getBiro() {
    return Biro;
}

public void setBiro(String Biro) {
    this.Biro = Biro;
}

public String getUgyek_szama() {
    return Ugyek_szama;
}

public void setUgyek_szama(String Ugyek_szama) {
    this.Ugyek_szama = Ugyek_szama;
}

public String getItelet() {
    return Itelet;
}

public void setItelet(String Itelet) {
```

```
        this.Itelet = Itelet;
    }
}

class Biro {
    private String biroId;
    private String biro_nev;
    private String tapasztalat;
    private String tanitvanyok;

    public Biro(String biroId, String biro_nev, String tapasztalat, String tanitvanyok) {
        this.biroId = biroId;
        this.biro_nev = biro_nev;
        this.tapasztalat = tapasztalat;
        this.tanitvanyok = tanitvanyok;
    }

    public String getBiroId() {
        return biroId;
    }

    public void setBiroId(String biroId) {
        this.biroId = biroId;
    }

    public String getBiro_nev() {
        return biro_nev;
    }

    public void setBiro_nev(String biro_nev) {
```

```
        this.biro_nev = biro_nev;
    }

    public String getTapasztalat() {
        return tapasztalat;
    }

    public void setTapasztalat(String tapasztalat) {
        this.tapasztalat = tapasztalat;
    }

    public String getTanitvanyok() {
        return tanitvanyok;
    }

    public void setTanitvanyok(String tanitvanyok) {
        this.tanitvanyok = tanitvanyok;
    }
}

class Jegyzo {
    private String jegyzo_Id;
    private String rogzites;
    private String jegyzo_nev;
    private String munkaIdo;
    private String nyelvtudas;

    public Jegyzo(String jegyzo_Id, String rogzites, String jegyzo_nev, String munkaIdo,
String nyelvtudas) {
        this.jegyzo_Id = jegyzo_Id;
        this.rogzites = rogzites;
```

```
    this.jegyzo_nev = jegyzo_nev;
    this.munkaIdo = munkaIdo;
    this.nyelvtudas = nyelvtudas;
}

public String getJegyzo_Id() {
    return jegyzo_Id;
}

public void setegyzo_Id(String jegyzoAzonosito) {
    this.jegyzo_Id = jegyzoAzonosito;
}

public String getRogzites() {
    return rogzites;
}

public void setRogzites(String rogzitesAzonosito) {
    this.rogzites = rogzitesAzonosito;
}

public String getJegyzo_nev() {
    return jegyzo_nev;
}

public void setJegyzo_nev(String nev) {
    this.jegyzo_nev = nev;
}

public String getMunkaIdo() {
```

```
        return munkaIdo;
    }

    public void setMunkaIdo(String munkaIdo) {
        this.munkaIdo = munkaIdo;
    }

    public String getNyelvtudas() {
        return nyelvtudas;
    }

    public void setNyelvtudas(String nyelvtudas) {
        this.nyelvtudas = nyelvtudas;
    }
}


public class DOMWriteZUYISF {

    public static void main(String[] args) {
        try {
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

            //XML fájl beolvasása.
            Document eredetiDoc = dBuilder.parse(new InputSource("XMLZUYISF.xml"));
            eredetiDoc.getDocumentElement().normalize();

            //Adatok előkészítése.
            List<Ugyved> ugyvedDataList = UgyvedXML(eredetiDoc);
```

```
        List<Vadlott> vadlottDataList = VadlottXML(eredetiDoc);
        List<Bunose> bunoseDataList = BunoseXML(eredetiDoc);
        List<Ugy> ugyDataList = UgyXML(eredetiDoc);
        List<Iteletethoz> IteletethozLista = IteletetHozXML(eredetiDoc);
        List<Biro> biroDataList = BiroXML(eredetiDoc);
        List<Jegyzo> jegyzoList = JegyzoXML(eredetiDoc);

        //Komment.
        List<String> kommentekList = getKommentek(eredetiDoc);

        //Új XML dokumentum létrehozása, ez lesz amivel kiírok.
        Document ujDoc = dBuilder.newDocument();

        //Felépíteni az XML fát a forrás dokumentum tartalmából.
        buildTree(ujDoc, ugyvedDataList, vadlottDataList, bunoseDataList, ugyDataList,
IteletethozLista, biroDataList, jegyzoList, kommentekList);

        //Menti az új dokumentumot az XMLZUYISF1.xml fájlba.
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4");

        DOMSource source = new DOMSource(ujDoc);
        StreamResult result = new StreamResult(new File("XMLZUYISF1.xml"));
        transformer.transform(source, result);

        //Kiírja a létrehozott XML fát a konzolra.
        Kiiras(ujDoc);

    } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }

    private static List<String> getKommentek(Document eredetiDoc) {
        List<String> kommentekList = new ArrayList<>();
        kommentgyujtes(eredetiDoc, kommentekList);
        return kommentekList;
    }

    private static void kommentgyujtes(Node node, List<String> kommentekList) {
        if (node.getNodeType() == Node.COMMENT_NODE) {
            kommentekList.add(((Comment) node).getTextContent());
        }

        NodeList children = node.getChildNodes();
        for (int i = 0; i < children.getLength(); i++) {
            kommentgyujtes(children.item(i), kommentekList);
        }
    }

    private static void Kiiras(Document doc) {
        try {
            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");

            StringWriter writer = new StringWriter();
            transformer.transform(new DOMSource(doc), new StreamResult(writer));
```

```
            //Kiírja az XML dokumentumot a konzolra.
            System.out.println(writer.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void buildTree(
            Document doc,
            List<Ugyved> ugyvedDataList,
            List<Vadlott> vadlottDataList,
            List<Bunose> bunoseDataList,
            List<Ugy> ugyDataList,
            List<Iteletethoz> Iteletethoz,
            List<Biro> biroDataList,
            List<Jegyzo> jegyzoList,
                    List<String> kommentekList) {

                        //Gyökér elem (Tárgyalás_ZUYISF) hozzáadása.
                        Element rootElement = doc.createElement("Tárgyalás_ZUYISF");
                        rootElement.setAttribute("xmlns:xs",
"http://www.w3.org/2001/XMLSchema-instance");
                        rootElement.setAttribute("xs:noNamespaceSchemaLocation",
"XMLSchemaZUYISF.xsd");

                        boolean[] Comment = { false, false, false, false, false, false, false };
//Boolean primitív típus, nem működik vele a referenciaátadás, ezért boolean tömb.

                        addUgyved(doc, rootElement, ugyvedDataList, kommentekList,
Comment);
```

```
                addVadlott(doc, rootElement, vadlottDataList, kommentekList,
Comment);
                addBunose(doc, rootElement, bunoseDataList, kommentekList,
Comment);
                addUgys(doc, rootElement, ugyDataList, kommentekList, Comment);
                addIteletethoz(doc, rootElement, Iteletethoz, kommentekList, Comment);
                addBiro(doc, rootElement, biroDataList, kommentekList, Comment);
                addJegyzo(doc, rootElement, jegyzoList, kommentekList, Comment);

                doc.appendChild(rootElement);

            }

    private static void addUgyved(Document doc, Element rootElement, List<Ugyved>
ugyvedDataList, List<String> kommentekList, boolean[] Comment) {
        for (Ugyved ugyvedData : ugyvedDataList) {
            addUgyved(doc, rootElement, ugyvedData, kommentekList, Comment);
        }
    }

    private static void addUgyved(Document doc, Element parentElement, Ugyved
ugyvedData, List<String> kommentekList, boolean[] Comment) {

        if (kommentekList.stream().anyMatch(comment -> comment.contains("Ügyvéd")) &&
!Comment[0]) {
            Comment commentNode = doc.createComment("Ügyvéd");
            parentElement.appendChild(commentNode);
            Comment[0] = true;
        }

            Element ugyvedElement = doc.createElement("Ügyvéd");
        ugyvedElement.setAttribute("Ügyvédi_ID", ugyvedData.getUgyvediId());
```

```
        ugyvedElement.setAttribute("Védi", ugyvedData.getVedi());

        Element szakteruletElement = doc.createElement("Szakterület");
        szakteruletElement.setTextContent(ugyvedData.getSzakterulet());
        ugyvedElement.appendChild(szakteruletElement);

        for (String telefonszam : ugyvedData.getTelefonszam()) {
            Element telefonszamElement = doc.createElement("Telefonszám");
            telefonszamElement.setTextContent(telefonszam);
            ugyvedElement.appendChild(telefonszamElement);
        }

        Element nevElement = doc.createElement("Ügyvéd_Név");
        nevElement.setTextContent(ugyvedData.getUgyved_nev());
        ugyvedElement.appendChild(nevElement);

        parentElement.appendChild(ugyvedElement);
    }

    private static void addVadlott(Document doc, Element rootElement, List<Vadlott>
vadlottDataList, List<String> kommentekList, boolean[] Comment) {
        for (Vadlott vadlott : vadlottDataList) {
            addVadlott(doc, rootElement, vadlott, kommentekList, Comment);
        }
    }

    private static void addVadlott(Document doc, Element parentElement, Vadlott vadlott,
List<String> kommentekList, boolean[] Comment) {

        if (kommentekList.stream().anyMatch(comment -> comment.contains("Vádlott")) &&
!Comment[1]) {
```

```
    Comment commentNode = doc.createComment("Vádlott");
    parentElement.appendChild(commentNode);
    Comment[1] = true;
}

Element vadlottElement = doc.createElement("Vádlott");
vadlottElement.setAttribute("Személyi_ig", vadlott.getSzemelyi_ig());

Element lakcimElement = doc.createElement("Lakcím");
lakcimElement.setTextContent(vadlott.getLakcim());
vadlottElement.appendChild(lakcimElement);

Element szulidoElement = doc.createElement("Szülidő");
szulidoElement.setTextContent(vadlott.getSzulido());
vadlottElement.appendChild(szulidoElement);

Element korElement = doc.createElement("Kor");
korElement.setTextContent(vadlott.getKor());
vadlottElement.appendChild(korElement);

Element priuszElement = doc.createElement("Priusz");
priuszElement.setTextContent(vadlott.getPriusz());
vadlottElement.appendChild(priuszElement);

Element nevElement = doc.createElement("Név");
Element vnevElement = doc.createElement("Vnév");
vnevElement.setTextContent(vadlott.getVnev());
Element knevElement = doc.createElement("Knév");
knevElement.setTextContent(vadlott.getKnev());
nevElement.appendChild(vnevElement);
```

```
        nevElement.appendChild(knevElement);
        vadlottElement.appendChild(nevElement);

        parentElement.appendChild(vadlottElement);
    }

    private static void addBunose(Document doc, Element rootElement, List<Bunose>
bunoseDataList, List<String> kommentekList, boolean[] Comment) {
        for (Bunose bunose : bunoseDataList) {
            addBunose(doc, rootElement, bunose, kommentekList, Comment);
        }
    }

    private static void addBunose(Document doc, Element parentElement, Bunose bunose,
List<String> kommentekList, boolean[] Comment) {

        if (kommentekList.stream().anyMatch(comment -> comment.contains("Bűnös-e")) &&
!Comment[2]) {
            Comment commentNode = doc.createComment("Bűnös-e");
            parentElement.appendChild(commentNode);
            Comment[2] = true;
        }

        Element bunoseElement = doc.createElement("Bűnös-e");
        bunoseElement.setAttribute("Vádlott", bunose.getVadlott());
        bunoseElement.setAttribute("Ügy", bunose.getUgy());

        Element vadElement = doc.createElement("Vád");
        vadElement.setTextContent(bunose.getVad());
        bunoseElement.appendChild(vadElement);
```

```
        parentElement.appendChild(bunoseElement);
    }

    private static void addUgys(Document doc, Element rootElement, List<Ugy> ugyDataList,
List<String> kommentekList, boolean[] Comment) {
        for (Ugy ugy : ugyDataList) {
            addUgy(doc, rootElement, ugy, kommentekList, Comment);
        };
    }

    private static void addUgy(Document doc, Element parentElement, Ugy ugy, List<String>
kommentekList, boolean[] Comment) {

        if (kommentekList.stream().anyMatch(comment -> comment.contains("Ügy")) &&
!Comment[3]) {
            Comment commentNode = doc.createComment("Ügy");
            parentElement.appendChild(commentNode);
            Comment[3] = true;
        }


        Element ugyElement = doc.createElement("Ügy");
        ugyElement.setAttribute("Ügyszám_ID", ugy.getUgyszamId());

        Element tipusElement = doc.createElement("Típus");
        tipusElement.setTextContent(ugy.getTipus());
        ugyElement.appendChild(tipusElement);

        Element dateElement = doc.createElement("Dátum");
        dateElement.setTextContent(ugy.getDatum());
        ugyElement.appendChild(dateElement);
```

```
        Element allapotElement = doc.createElement("Állapot");
        allapotElement.setTextContent(ugy.getAllapot());
        ugyElement.appendChild(allapotElement);

        parentElement.appendChild(ugyElement);
    }

    private static void addIteletethoz(Document doc, Element rootElement, List<Iteletethoz>
iteletethozList, List<String> kommentekList, boolean[] Comment) {
        for (Iteletethoz iteletethoz : iteletethozList) {
            addIteletethoz(doc, rootElement, iteletethoz, kommentekList, Comment);
        }
    }

    private static void addIteletethoz(Document doc, Element parentElement, Iteletethoz
iteletethoz, List<String> kommentekList, boolean[] Comment) {

        if (kommentekList.stream().anyMatch(comment -> comment.contains("Ítéletet_hoz"))
&& !Comment[4]) {
            Comment commentNode = doc.createComment("Ítéletet_hoz");
            parentElement.appendChild(commentNode);
            Comment[4] = true;
        }

        Element iteletethozElement = doc.createElement("Ítéletet_hoz");
        iteletethozElement.setAttribute("Ügy", iteletethoz.getUgy());
        iteletethozElement.setAttribute("Bíró", iteletethoz.getBiro());

        Element ugyekszamaElement = doc.createElement("Ügyek_száma");
        ugyekszamaElement.setTextContent(String.valueOf(iteletethoz.getUgyek_szama()));
```

```
        iteletethozElement.appendChild(ugyekszamaElement);

        Element iteletElement = doc.createElement("Ítélet");
        iteletElement.setTextContent(iteletethoz.getItelet());
        iteletethozElement.appendChild(iteletElement);

        parentElement.appendChild(iteletethozElement);
    }

    private static void addBiro(Document doc, Element rootElement, List<Biro> biroDataList,
List<String> kommentekList, boolean[] Comment) {
        for (Biro biro : biroDataList) {
            addBiro(doc, rootElement, biro, kommentekList, Comment);
        }
    }

    private static void addBiro(Document doc, Element parentElement, Biro biro, List<String>
kommentekList, boolean[] Comment) {

        if (kommentekList.stream().anyMatch(comment -> comment.contains("Bíró")) &&
!Comment[5]) {
            Comment commentNode = doc.createComment("Bíró");
            parentElement.appendChild(commentNode);
            Comment[5] = true;
        }

        Element biroElement = doc.createElement("Bíró");
        biroElement.setAttribute("Bíró_ID", biro.getBiroId());

        Element nevElement = doc.createElement("Bíró_Név");
        nevElement.setTextContent(biro.getBiro_nev());
```

```
        biroElement.appendChild(nevElement);

        Element tapasztalatokElement = doc.createElement("Tapasztalat");
        tapasztalatokElement.setTextContent(biro.getTapasztalat());
        biroElement.appendChild(tapasztalatokElement);

        Element tanitvanyokElement = doc.createElement("Tanítványok");
        tanitvanyokElement.setTextContent(biro.getTanitvanyok());
        biroElement.appendChild(tanitvanyokElement);

        parentElement.appendChild(biroElement);
    }

    private static void addJegyzo(Document doc, Element rootElement, List<Jegyzo>
jegyzoList, List<String> kommentekList, boolean[] Comment) {
        for (Jegyzo jegyzo : jegyzoList) {
            addJegyzo(doc, rootElement, jegyzo, kommentekList, Comment);
        }
    }

    private static void addJegyzo(Document doc, Element parentElement, Jegyzo jegyzo,
List<String> kommentekList, boolean[] Comment) {

        if (kommentekList.stream().anyMatch(comment -> comment.contains("Jegyzo")) &&
!Comment[6]) {
            Comment commentNode = doc.createComment("Jegyzo");
            parentElement.appendChild(commentNode);
            Comment[6] = true;
        }

        Element jegyzoElement = doc.createElement("Jegyző");
```

```
        jegyzoElement.setAttribute("Jegyző_ID", jegyzo.getJegyzo_Id());
        jegyzoElement.setAttribute("Rögzítés", jegyzo.getRogzites());

        Element nevElement = doc.createElement("Jegyző_Név");
        nevElement.setTextContent(jegyzo.getJegyzo_nev());
        jegyzoElement.appendChild(nevElement);

        Element munkaidoElement = doc.createElement("Munkaidő");
        munkaidoElement.setTextContent(jegyzo.getMunkaIdo());
        jegyzoElement.appendChild(munkaidoElement);

        Element nyelvtudasElement = doc.createElement("Nyelvtudás");
        nyelvtudasElement.setTextContent(String.join(", ", jegyzo.getNyelvtudas()));
        jegyzoElement.appendChild(nyelvtudasElement);

        parentElement.appendChild(jegyzoElement);
    }

    private static List<Ugyved> UgyvedXML(Document doc) {
        List<Ugyved> UgyvedList = new ArrayList<>();
        NodeList nodeList = doc.getElementsByTagName("Ügyvéd");

        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                String vedi = element.getAttribute("Védi");
                String ugyvediId = element.getAttribute("Ügyvédi_ID");
                String szakterulet =
element.getElementsByTagName("Szakterület").item(0).getTextContent();
```

```
            NodeList telefonszamNodes = element.getElementsByTagName("Telefonszám");
//Többértékű.
            List<String> telefonszamList = new ArrayList<>();
            for (int j = 0; j < telefonszamNodes.getLength(); j++) {
               Node telefonszamNode = telefonszamNodes.item(j);
               if (telefonszamNode.getNodeType() == Node.ELEMENT_NODE) {
                  telefonszamList.add(telefonszamNode.getTextContent());
               }
            }

            String ugyved_nev =
element.getElementsByTagName("Ügyvéd_Név").item(0).getTextContent();

            //Ugyved objektum létrehozása és hozzáadása a listához.
            UgyvedList.add(new Ugyved(vedi, ugyvediId, szakterulet, telefonszamList,
ugyved_nev));
         }
      }
      return UgyvedList;
   }

   private static List<Vadlott> VadlottXML(Document doc) {
      List<Vadlott> VadlottList = new ArrayList<>();
      NodeList nodeList = doc.getElementsByTagName("Vádlott");

      for (int i = 0; i < nodeList.getLength(); i++) {
         Node node = nodeList.item(i);
         if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;
```

```
            String szemelyi_ig = element.getAttribute("Személyi_ig");
            String lakcim =
element.getElementsByTagName("Lakcím").item(0).getTextContent();
            String szulido =
element.getElementsByTagName("Szülidő").item(0).getTextContent();
            String kor = element.getElementsByTagName("Kor").item(0).getTextContent();
            String priusz =
element.getElementsByTagName("Priusz").item(0).getTextContent();
            String vnev = element.getElementsByTagName("Vnév").item(0).getTextContent();
            String knev = element.getElementsByTagName("Knév").item(0).getTextContent();

            //Vadlott objektum létrehozása és hozzáadása a listához.
            VadlottList.add(new Vadlott(szemelyi_ig, lakcim, szulido, kor, priusz, vnev,
knev));
        }
      }
      return VadlottList;
   }

   private static List<Bunose> BunoseXML(Document doc) {
      List<Bunose> BunoseList = new ArrayList<>();
      NodeList nodeList = doc.getElementsByTagName("Bűnös-e");

      for (int i = 0; i < nodeList.getLength(); i++) {
         Node node = nodeList.item(i);
         if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;

            String vadlott = element.getAttribute("Vádlott");
            String ugy = element.getAttribute("Ügy");
            String vad = element.getElementsByTagName("Vád").item(0).getTextContent();
```

```
            //Bunose objektum létrehozása és hozzáadása a listához.
            BunoseList.add(new Bunose(vadlott, ugy, vad));
        }
    }
    return BunoseList;
}

private static List<Ugy> UgyXML(Document doc) {
    List<Ugy> UgyList = new ArrayList<>();
    NodeList nodeList = doc.getElementsByTagName("Ügy");

    for (int i = 0; i < nodeList.getLength(); i++) {
        Node node = nodeList.item(i);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;

            String ugy = element.getAttribute("Ügyszám_ID");
            String tipus = element.getElementsByTagName("Típus").item(0).getTextContent();
            String datum =
element.getElementsByTagName("Dátum").item(0).getTextContent();
            String allapot =
element.getElementsByTagName("Állapot").item(0).getTextContent();

            //Ugy objektum létrehozása és hozzáadása a listához.
            UgyList.add(new Ugy(ugy, tipus, datum, allapot));
        }
    }
    return UgyList;
}
```

```
    private static List<Iteletethoz> IteletetHozXML(Document doc) {
        List<Iteletethoz> IteletetHozList = new ArrayList<>();
        NodeList nodeList = doc.getElementsByTagName("Ítéletet_hoz");

        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;

                String ugy = element.getAttribute("Ügy");
                String biro = element.getAttribute("Bíró");
                String ugyek_szama =
element.getElementsByTagName("Ügyek_száma").item(0).getTextContent();
                String itelet = element.getElementsByTagName("Ítélet").item(0).getTextContent();

                //Iteletethoz objektum létrehozása és hozzáadása a listához.
                IteletetHozList.add(new Iteletethoz(ugy, biro, ugyek_szama, itelet));
            }
        }
        return IteletetHozList;
    }

    private static List<Biro> BiroXML(Document doc) {
        List<Biro> BiroDataList = new ArrayList<>();
        NodeList nodeList = doc.getElementsByTagName("Bíró");

        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;
```

```
            String biro_Id = element.getAttribute("Bíró_ID");
            String biro_nev =
element.getElementsByTagName("Bíró_Név").item(0).getTextContent();
            String tapasztalat =
element.getElementsByTagName("Tapasztalat").item(0).getTextContent();
            String tanitvanyok =
element.getElementsByTagName("Tanítványok").item(0).getTextContent();

            //Biro objektum létrehozása és hozzáadása a listához.
            BiroDataList.add(new Biro(biro_Id, biro_nev, tapasztalat, tanitvanyok));
        }
      }
      return BiroDataList;
   }


   private static List<Jegyzo> JegyzoXML(Document doc) {
      List<Jegyzo> jegyzoList = new ArrayList<>();
      NodeList nodeList = doc.getElementsByTagName("Jegyző");

      for (int i = 0; i < nodeList.getLength(); i++) {
         Node node = nodeList.item(i);
         if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;

            String jegyzo_Id = element.getAttribute("Jegyző_ID");
            String rogzites = element.getAttribute("Rögzítés");
            String jegyzo_nev =
element.getElementsByTagName("Jegyző_Név").item(0).getTextContent();
            String munkaido =
element.getElementsByTagName("Munkaidő").item(0).getTextContent();
```

```
            String nyelvtudas =
element.getElementsByTagName("Nyelvtudás").item(0).getTextContent();


            //Jegyzo objektum létrehozása és hozzáadása a listához.
            jegyzoList.add(new Jegyzo(jegyzo_Id, rogzites, jegyzo_nev, munkaido,
nyelvtudas));
        }
      }
      return jegyzoList;
   }
}
```

Dr. Bednarik László tárgyjegyző