JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Tárgyalás

Készítette: **Juhász Balázs**

Neptunkód: **ZUYISF**

Dátum: 2023.11.14

Tartalomjegyzék

Tartalom

Bevezetés

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

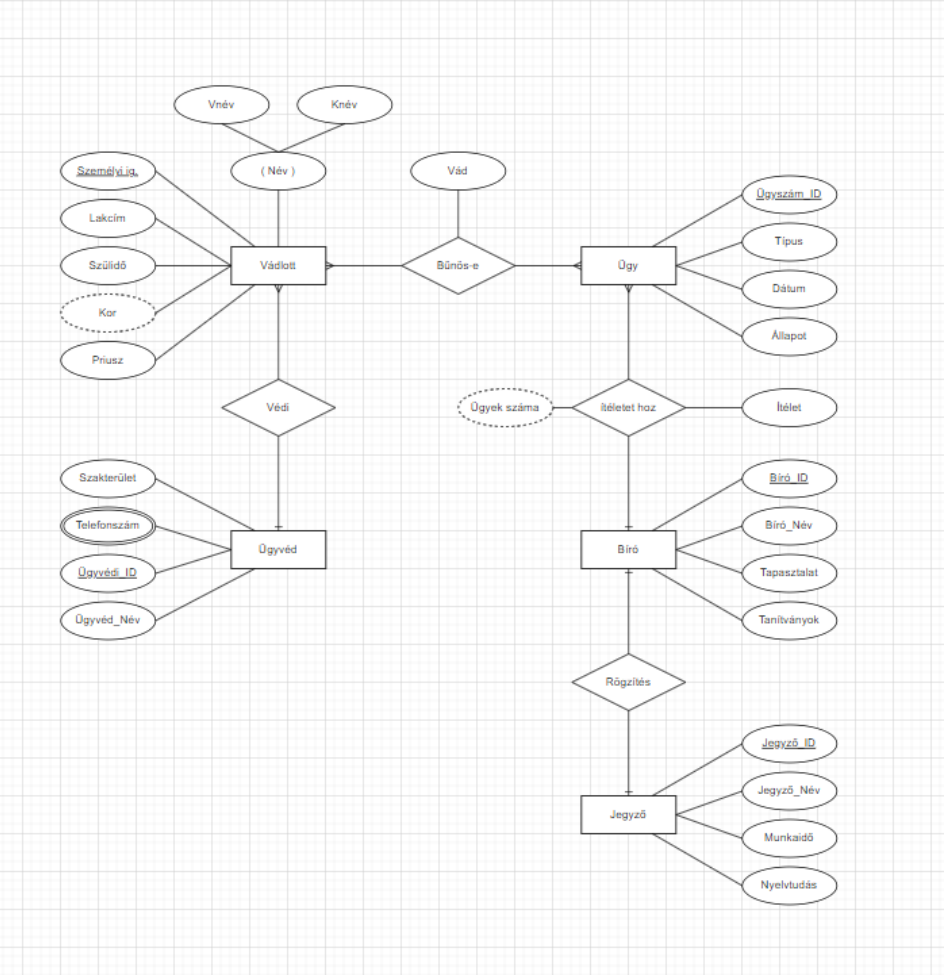
Az általam elkészített adatbázis egy bírósági tárgyalásokat nyilvántartó adatbázis adatmodelljét ábrázolja. A téma a valóságához közel áll, erre nagyon nagy szükség van a világot tekintve bárhol. Az adatbázisban öt egyed található: Vádlott, Ügy, Ügyvéd, Bíró és Jegyző. A Vádlott egyedből nyílik 6 darab 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

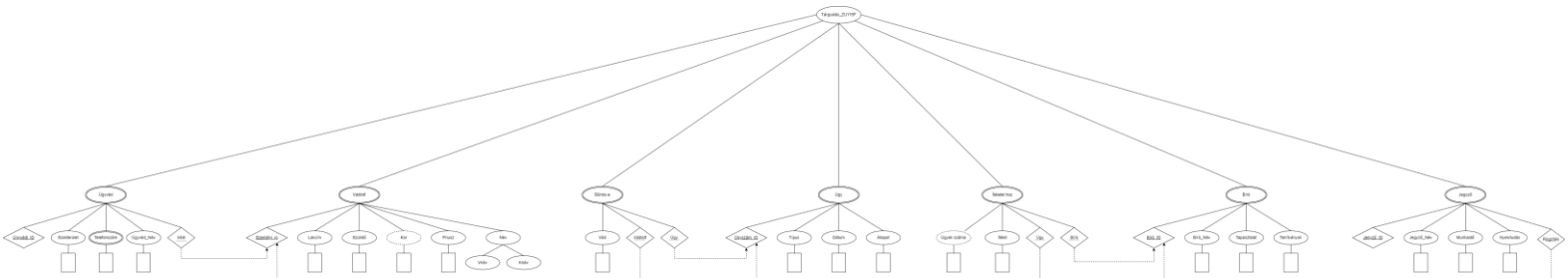
1a) Az adatbázis ER modell tervezése (Legyen legalább 5 egyed, többféle kapcsolat (1:1; 1:N; M:N), tulajdonságok - normál, kulcs, összetett, többértékű. (Csak szerkesztő programmal rajzolt ábra megfelelő, szabványos szimbólumok használata) megvalósítás rövid leírás…., majd a modell

A Tárgyalás ER modell 5 egyedből áll, mindegyik egyednek van legalább 4 tulajdonsága. Mindegyik egyed kapcsolatban van egy másik egyeddel, 1-Több, Több-Több, 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.liquid-technologies.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>

<!--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>

<!--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>

<!--Ügy-->
<Ügy Ügyszám_ID="3000">
```

<Típus>Büntetőjogi eljárás</Típus>

<Dátum>2023-11-09</Dátum>

<Állapot>Folyamatban</Állapot>

</Ügy>

<Ügy Ügyszám_ID="3001">

<Típus>Szabálysértési eljárás</Típus>

<Dátum>2023-08-16</Dátum>

<Állapot>Elbírálás alatt</Állapot>

</Ügy>

<!--Ítéletet_hoz-->

<Ítéletet_hoz Ügy="3000" Bíró="1000">

<Ügyek_száma>6</Ügyek_száma>

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

</Ítéletet_hoz>

<Ítéletet_hoz Ügy="3001" Bíró="1000">

<Ügyek_száma>4</Ügyek_száma>

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

```
    </Ítéletet_hoz>

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

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


</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="xs:string" />
    <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="xs:int" /> <!-- 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="1"/>
        <xs:element name="Jegyző" type="JegyzőTípus" minOccurs="1"
maxOccurs="1"/>
      </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>




</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 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.stream.events.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.SAXException;
import javax.xml.parsers.ParserConfigurationException;
import java.io.IOException;



public class DomReadZUYISF {

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

try
{

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

        System.out.println("Gyökérelem: " + doc.getDocumentElement().getNodeName());
```

```
    NodeList nodeList = doc.getElementsByTagName("Ügyvéd"); //Ügyvéd elemek.
    printNodeList(nodeList);

    nodeList = doc.getElementsByTagName("Vádlott"); //Vádlott elemek.
    printNodeList(nodeList);

    nodeList = doc.getElementsByTagName("Bűnös-e"); //Bűnös-e elemek.
    printNodeList(nodeList);

    nodeList = doc.getElementsByTagName("Ügy"); //Ügy elemek.
    printNodeList(nodeList);

    nodeList = doc.getElementsByTagName("Ítéletet_hoz"); //Ítéletet_hoz elemek.
    printNodeList(nodeList);

    nodeList = doc.getElementsByTagName("Bíró"); //Bíró elemek.
    printNodeList(nodeList);

    nodeList = doc.getElementsByTagName("Jegyző"); //Jegyző elemek.
    printNodeList(nodeList);

    Element root = doc.getDocumentElement();
    System.out.println("Gyökérelem: " + root.getNodeName()); //Gyökérelem.

}
catch (Exception e)
{
```

```
        e.printStackTrace();
}
}

private static void printNodeList(NodeList nodeList) {
for (int temp = 0; temp < nodeList.getLength(); temp++)
{
Node nNode = nodeList.item(temp);
    System.out.println("\nJelenlegi elem: " + nNode.getNodeName());

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

            System.out.println("Comment: " + nNode.getTextContent());  // Kiírja a komment
tartalmát
            continue;

    }


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

            Element eElement = (Element) nNode;
            System.out.println("Elemhez tartozó attribútumok: ");
        for (int j = 0; j < eElement.getAttributes().getLength(); j++)
        {
            Node attr = eElement.getAttributes().item(j);
```

```
            System.out.println(" - " + attr.getNodeName() + ": " + attr.getNodeValue());
        }

        NodeList children = eElement.getChildNodes();
        System.out.println("Gyerek elemei:");
        for (int j = 0; j < children.getLength(); j++)
        {

           Node child = children.item(j);

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

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

           }
        }
    }
}
}
}
```

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(elementId) ||
                (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(elementId)
&&
                        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 javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
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;

public class DOMWriteZUYISF {

   public static void main(String[] args)
   {

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

         printNode(doc.getDocumentElement(), 0); //Fa struktúra

         TransformerFactory transformerFactory = TransformerFactory.newInstance();
         Transformer transformer = transformerFactory.newTransformer();
         transformer.setOutputProperty(OutputKeys.INDENT, "yes"); // Bekezdések
hozzáadása
```

```
        transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "4"); // Indentálási mélység

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

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

  private static void printNode(Node node, int depth)
  {

    String indent = " ".repeat(depth * 4);

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

        System.out.println(indent + "<" + node.getNodeName() + ">");
        NodeList nodeList = node.getChildNodes();

        for (int i = 0; i < nodeList.getLength(); i++)
        {
           printNode(nodeList.item(i), depth + 1);
        }
        System.out.println(indent + "</" + node.getNodeName() + ">");
    }
```

```
        else if (node.getNodeType() == Node.TEXT_NODE &&
!node.getTextContent().trim().isEmpty())
        {
            System.out.println(indent + node.getTextContent().trim());
        }


    }
}
```

Dr. Bednarik László

tárgyjegyző