

JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Iskolai adatbázis

Készítette: **Kórád György**

Neptunkód: **ZF440N**

Dátum: 2023.12.01

1a) A feladat témája

A beadandó feladatom témája egy iskolai adatbázis, amely nyilván tartja a tanárok és tanulók adatait, valamint a tanulók megszerzett értékeléseit tantárgyak szerint. A tanulókat osztályok szerint is megtalálhatjuk, illetve azok osztályfőnökét.

Az ER modell egyedei és tulajdonságai:

- A diák egyed
 - Név: a tanuló neve
 - Lakcím: a tanuló lakcíme
 - Születési idő: a tanuló születési ideje
 - Anyja neve: a tanuló anyja neve
 - Kor: a tanuló életkora

- A tanár egyed
 - Név: a tanár neve
 - Lakcím: a tanár lakcíme
 - Születési idő: a tanár születési ideje
 - telefonszám: a tanár telefonszáma

- Az osztály egyed
 - Létszám: tanulók száma az osztályban
 - Tagozat: például matematika vagy testnevelés

- A tantárgy egyed:
 - Név: a tantárgy neve

- A tankönyv egyed
 - Cím: a tankönyv címe
 - Író: a tankönyv írója
 - Kiadó: a tankönyv kiadója
 - Kiadás ideje: a tankönyv kiadási ideje

Az egyedek közötti kapcsolatok:

Diák és osztály:

A diák és az osztály egyedek között egy a többhöz kapcsolat van, mert egy osztályhoz több diák tartozik, de egy diák nem járhat több osztályba egyszerre.

Tanár és osztály:

A tanár és az osztály között egy-egy kapcsolat van, mert egy tanár csak egy osztályban lehet osztályfőnök.

Tanár és tantárgy:

Ezek között több-több kapcsolat van, mivel egy tárgyat több tanár is taníthat, valamit egy tanár több tárgyat is taníthat. A kapcsolat paraméterei szint és mióta.

- Szint: például emelt szint
- Mióta: az a dátum amióta egy adott tanár tanítja a tárgyat.

Diák és tantárgy:

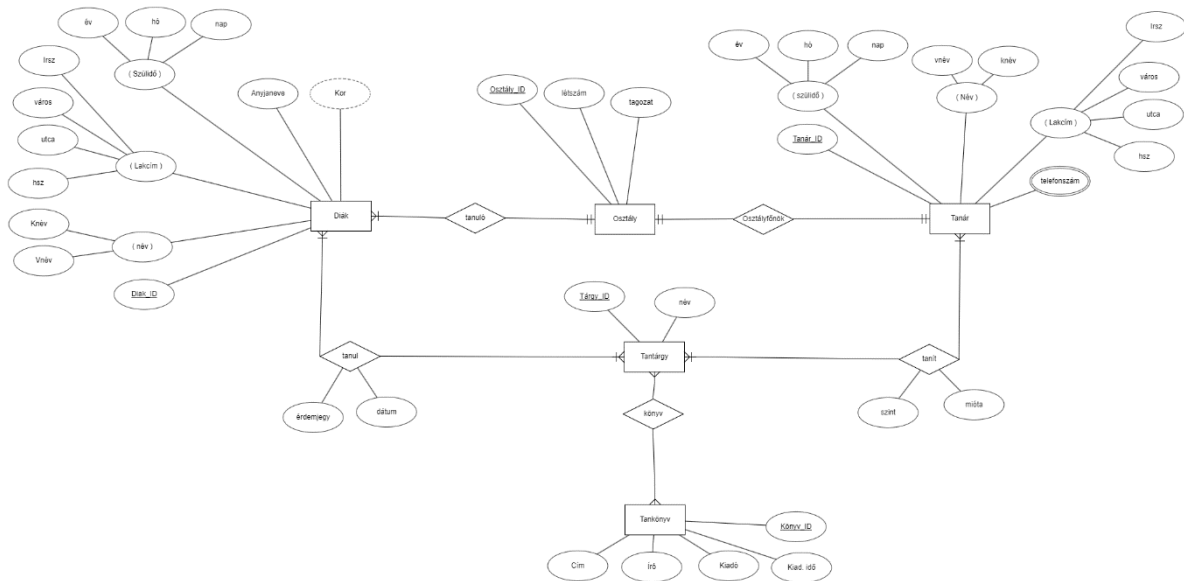
Itt is több-több kapcsolat van, mert egy diák több tantárgyat tanul, és egy tantárgyhoz több diák tartozik. A kapcsolat paraméterei érdemjegy és dátum.

- Érdemjegy: a tanuló megszerzett érdemjegye adott tantárgyból
- Dátum: az érdemjegy megszerzésének dátuma

Tantárgy és tankönyv:

Itt is több-több kapcsolat van, viszont ez egy nem kötelező kapcsolat. Vannak olyan tantárgyak ahová több tankönyv tartozik, de van olyan is ahová egy sem.

A feladat ER modellje:



Az egyedek közötti kapcsolat

- Tanuló kapcsolat: Egy osztályhoz több tanuló tartozik 1:N
- Osztályfőnök kapcsolat: Egy osztályban egy tanár az osztályfőnök 1:1
- Tanít kapcsolat: Több tanár több tantárgyat tanít N:N
- Tanul kapcsolat: Több tanuló több tantárgyat tanul N:N
- Könyv kapcsolat: Több tantárgyhoz több tankönyv is tartozhat N:N

1b) Az ER-modell konvertálása XDM modellre

Az XDM (XML Data Model) modell esetén a struktúrát XML elemek és attribútumok segítségével ábrázoljuk. Az ábrázolásban használt jelölések, mint az ellipszis, rombusz, téglalap és nyilak, segítenek a különböző típusú entitások, attribútumok és kapcsolatok megkülönböztetésében

Ellipszis (Elem):

Az XML elemeket ellipszis jelöli. Ezek az elemek lehetnek egyedek vagy más elemek gyűjteményei.

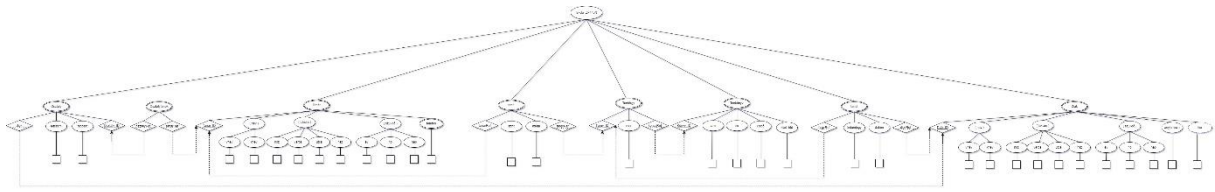
Rombusz (Attribútum):

Az attribútumokat rombusz jelöli. Az attribútumok a kulcsokból származhatnak.

Téglalap (Szöveg):

A szövegek vagy karakterláncokat téglalap jelöli. Ezek a szövegek fogják megjeleníteni az XML dokumentumban.

A feladat XDM modellje:



1c) XML dokumentum készítése

Az XDM modell alapján elkészítettem az XML dokumentumot. A root element az Iskola_ZF440N volt. A gyerek elemekből 4-4 példányt létrehoztam, ezek mind kulcsot kaptak és a hozzájuk társuló idegen kulcsot is, ha volt ilyen.

Az XML dokumentum forráskódja:

```
<?xml version="1.0" encoding="UTF-8"?>
<Iskola_ZF440N xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="XMLSchemaZF440N.xsd">

  <!--Tanárok-->

  <Tanar tanarID="1">
    <nev>
      <vnev>Kovács</vnev>
      <knev>Anna</knev>
    </nev>
    <lakcim>
      <irsz>1024</irsz>
      <varos>Budapest</varos>
      <utca>Kossuth utca</utca>
      <hsz>5</hsz>
    </lakcim>
    <szulido>
      <ev>1985</ev>
      <ho>3</ho>
      <nap>10</nap>
    </szulido>
    <telefon>123456789</telefon>
  </Tanar>

  <Tanit tanitID="1" tanarRef="1" targyRef="1">
    <szint>közép</szint>
    <miota>2015</miota>
  </Tanit>
```

```
<Tanar tanarID="2">
  <nev>
    <vnev>Szabó</vnev>
    <knev>Péter</knev>
  </nev>
  <lakcim>
    <irsz>1132</irsz>
    <varos>Budapest</varos>
    <utca>Alkotás utca</utca>
    <hsz>12</hsz>
  </lakcim>
  <szulido>
    <ev>1978</ev>
    <ho>6</ho>
    <nap>25</nap>
  </szulido>
  <telefon>987654321</telefon>
</Tanar>

<Tanit tanitID="2" tanarRef="2" targyRef="2">
  <szint>alsó</szint>
  <miota>2010</miota>
</Tanit>

<Tanar tanarID="3">
  <nev>
    <vnev>Király</vnev>
    <knev>Mária</knev>
  </nev>
  <lakcim>
    <irsz>1098</irsz>
    <varos>Budapest</varos>
    <utca>Szent István utca</utca>
    <hsz>8</hsz>
  </lakcim>
  <szulido>
    <ev>1990</ev>
    <ho>12</ho>
    <nap>15</nap>
  </szulido>
  <telefon>555444333</telefon>
</Tanar>

<Tanit tanitID="3" tanarRef="3" targyRef="3">
  <szint>felső</szint>
  <miota>2012</miota>
</Tanit>

<Tanar tanarID="4">
```

```
<nev>
  <vnev>Nagy</vnev>
  <knev>Gábor</knev>
</nev>
<lakcim>
  <irsz>1145</irsz>
  <varos>Budapest</varos>
  <utca>Kossuth Lajos utca</utca>
  <hsz>20</hsz>
</lakcim>
<szulido>
  <ev>1982</ev>
  <ho>8</ho>
  <nap>5</nap>
</szulido>
<telefon>111222333</telefon>
</Tanar>

<Tanit tanitID="4" tanarRef="4" targyRef="4">
  <szint>közép</szint>
  <miota>2018</miota>
</Tanit>

<Tanar tanarID="5">
  <nev>
    <vnev>Kiss</vnev>
    <knev>Éva</knev>
  </nev>
  <lakcim>
    <irsz>1066</irsz>
    <varos>Budapest</varos>
    <utca>Andrássy út</utca>
    <hsz>18</hsz>
  </lakcim>
  <szulido>
    <ev>1988</ev>
    <ho>5</ho>
    <nap>20</nap>
  </szulido>
  <telefon>999888777</telefon>
</Tanar>

<Tanit tanitID="5" tanarRef="5" targyRef="5">
  <szint>felső</szint>
  <miota>2016</miota>
</Tanit>

<!--Diákok-->
```

```
<Diak diakID="1">
  <nev>
    <vnev>Nagy</vnev>
    <knev>Péter</knev>
  </nev>
  <lakcim>
    <irsz>1111</irsz>
    <varos>Szeged</varos>
    <utca>Petőfi utca</utca>
    <hsz>15</hsz>
  </lakcim>
  <szulido>
    <ev>2005</ev>
    <ho>7</ho>
    <nap>5</nap>
  </szulido>
  <anyjaneve>Nagy Mária</anyjaneve>
  <kor>18</kor>
</Diak>

<Tanul tanulID="1" diakRef="1" konyvRef="1">
  <erdemjegy>4</erdemjegy>
  <datum>2023-11-15</datum>
</Tanul>

<Diak diakID="2">
  <nev>
    <vnev>Kis</vnev>
    <knev>Anna</knev>
  </nev>
  <lakcim>
    <irsz>1020</irsz>
    <varos>Budapest</varos>
    <utca>Alkotmány utca</utca>
    <hsz>8</hsz>
  </lakcim>
  <szulido>
    <ev>2004</ev>
    <ho>5</ho>
    <nap>12</nap>
  </szulido>
  <anyjaneve>Kis Zsuzsanna</anyjaneve>
  <kor>19</kor>
</Diak>

<Tanul tanulID="2" diakRef="2" konyvRef="2">
  <erdemjegy>5</erdemjegy>
  <datum>2023-10-20</datum>
```



```
</Tanul>

<Diak diakID="3">
  <nev>
    <vnev>Tóth</vnev>
    <knev>Géza</knev>
  </nev>
  <lakcim>
    <irsz>1138</irsz>
    <varos>Budapest</varos>
    <utca>Váci utca</utca>
    <hsz>10</hsz>
  </lakcim>
  <szulido>
    <ev>2006</ev>
    <ho>8</ho>
    <nap>28</nap>
  </szulido>
  <anyjaneve>Tóth Éva</anyjaneve>
  <kor>17</kor>
</Diak>

<Tanul tanulID="3" diakRef="3" konyvRef="3">
  <erdemjegy>3</erdemjegy>
  <datum>2023-11-05</datum>
</Tanul>

<Diak diakID="4">
  <nev>
    <vnev>Nagy</vnev>
    <knev>Mária</knev>
  </nev>
  <lakcim>
    <irsz>1095</irsz>
    <varos>Budapest</varos>
    <utca>Kerepesi út</utca>
    <hsz>22</hsz>
  </lakcim>
  <szulido>
    <ev>2003</ev>
    <ho>4</ho>
    <nap>15</nap>
  </szulido>
  <anyjaneve>Nagy Erzsébet</anyjaneve>
  <kor>19</kor>
</Diak>

<Tanul tanulID="4" diakRef="4" konyvRef="4">
  <erdemjegy>5</erdemjegy>
```

```
<datum>2023-10-10</datum>
</Tanul>

<!--Osztályok-->

<Osztaly osztalyID = "1" >
  <letszam>30</letszam>
  <tagozat>Matematika</tagozat>
</Osztaly>

<Osztaly osztalyID = "2" >
  <letszam>24</letszam>
  <tagozat>Angol</tagozat>
</Osztaly>

<Osztaly osztalyID = "3" >
  <letszam>15</letszam>
  <tagozat>Testnevelés</tagozat>
</Osztaly>

<Tanulo osztalyRef = "1" diakRef = "1"/>
<Tanulo osztalyRef = "1" diakRef = "3"/>
<Tanulo osztalyRef = "2" diakRef = "4"/>
<Tanulo osztalyRef = "3" diakRef = "2"/>

<Osztalyfonok ofID = "1" tanarRef = "2" osztalyRef = "1"></Osztalyfonok>
<Osztalyfonok ofID = "2" tanarRef = "2" osztalyRef = "3"></Osztalyfonok>
<Osztalyfonok ofID = "3" tanarRef = "2" osztalyRef = "2"></Osztalyfonok>

<!--Tantárgyak és tankönyvek-->

<Tantargy targyID = "1">
  <nev>Magyar nyelv és irodalom</nev>
</Tantargy>

<Tankonyv TkonyvID = "1">
  <cim>Az elveszett idő nyomában</cim>
  <iro>József Attila</iro>
  <kiado>Könyvkiadó Kft.</kiado>
  <kiadido>2022-09-01</kiadido>
</Tankonyv>

<Konyv TkonyvRef = "1" targyRef = "1"/>

<Tantargy targyID="2">
  <nev>Matematika</nev>
</Tantargy>
```

```
<Tankonyv TkonyvID="2">
  <cim>Az algebrától a geometriáig</cim>
  <iro>Kovács István</iro>
  <kiado>Matektudás Kiadó</kiado>
  <kiadido>2023-03-15</kiadido>
</Tankonyv>

<Konyv TkonyvRef="2" targyRef="2"/>

<Tantargy targyID="3">
  <nev>Biológia</nev>
</Tantargy>

<Tankonyv TkonyvID="3">
  <cim>Az élővilág csodái</cim>
  <iro>Nagy Éva</iro>
  <kiado>Biokutató Könyvkiadó</kiado>
  <kiadido>2023-05-20</kiadido>
</Tankonyv>

<Konyv TkonyvRef="3" targyRef="3"/>

<Tantargy targyID="4">
  <nev>Történelem</nev>
</Tantargy>

<Tankonyv TkonyvID="4">
  <cim>Az emberiség története</cim>
  <iro>Kiss Gábor</iro>
  <kiado>Történelem Mester Kiadó</kiado>
  <kiadido>2023-07-10</kiadido>
</Tankonyv>

<Konyv TkonyvRef="4" targyRef="4"/>

<Tantargy targyID="5">
  <nev>Fizika</nev>
</Tantargy>

<Tankonyv TkonyvID="5">
  <cim>Az erők és mozgás törvényei</cim>
  <iro>Varga János</iro>
  <kiado>Fizika Guru Kiadó</kiado>
  <kiadido>2023-08-25</kiadido>
</Tankonyv>

<Konyv TkonyvRef="5" targyRef="5"/>

</Iskola_ZF440N>
```

1d) XML Schema készítés

Az XML Schemád részletesen meghatározza az "Iskola_ZF440N" gyökérelemhez kapcsolódó összes típust és azok attribútumait. Az egyszerű típusok, például a "NevType", "CimType" és "DatumType", részletesen leírják a nevek, címek és dátumok struktúráját. Az összetett típusok, mint például a "TanarType", "TanitType", "OsztalyType", "TantargyType" ezeket az egyszerű típusokat használják fel egyedi struktúrák létrehozásához.

Az XML Schemában az attribútumok típusai is pontosan meghatározottak, például az "osztalyID" vagy "tanarID" típusa "xs:string", míg az "erdemjegy" típusa "xs:string".

A gyökérelem típusa egy összetett típus, amelynek struktúrája egy választási lehetőségből áll, és ezáltal többféle típust képes kezelni. Ez a gyökérelem tartalmazhat többféle alárendelt elemet.

Az XMLSchema forráskódja:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- Egyszerű típusok -->

  <xs:complexType name="NevType">
    <xs:sequence>
      <xs:element name="vnev" type="xs:string"/>
      <xs:element name="knev" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="CimType">
    <xs:sequence>
      <xs:element name="irsz" type="xs:string"/>
      <xs:element name="varos" type="xs:string"/>
      <xs:element name="utca" type="xs:string"/>
      <xs:element name="hsz" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="DatumType">
    <xs:sequence>
      <xs:element name="ev" type="xs:integer"/>
      <xs:element name="ho" type="xs:integer"/>
      <xs:element name="nap" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="TanarType">
```

```

    <xs:sequence>
      <xs:element name="nev" type="NevType"/>
      <xs:element name="lakcim" type="CimType"/>
      <xs:element name="szulido" type="DatumType"/>
      <xs:element name="telefon" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="tanarID" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="TanitType">
    <xs:sequence>
      <xs:element name="szint" type="xs:string"/>
      <xs:element name="miota" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="tanitID" type="xs:string"/>
    <xs:attribute name="tanarRef" type="xs:string"/>
    <xs:attribute name="targyRef" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="OsztalyType">
    <xs:sequence>
      <xs:element name="letszam" type="xs:integer"/>
      <xs:element name="tagozat" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="osztalyID" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="TantargyType">
    <xs:sequence>
      <xs:element name="nev" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="targyID" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="TankonyvType">
    <xs:sequence>
      <xs:element name="cim" type="xs:string"/>
      <xs:element name="iro" type="xs:string"/>
      <xs:element name="kiado" type="xs:string"/>
      <xs:element name="kiadido" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="TkonyvID" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="DiakType">
    <xs:sequence>
      <xs:element name="nev" type="NevType"/>
      <xs:element name="lakcim" type="CimType"/>
      <xs:element name="szulido" type="DatumType"/>

```

```

        <xs:element name="anyjaneve" type="xs:string"/>
        <xs:element name="kor" type="xs:integer"/>
    </xs:sequence>
    <xs:attribute name="diakID" type="xs:string"/>
</xs:complexType>

<xs:complexType name="TanulType">
    <xs:sequence>
        <xs:element name="erdemjegy" type="xs:string"/>
        <xs:element name="datum" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="tanulID" type="xs:string"/>
    <xs:attribute name="diakRef" type="xs:string"/>
    <xs:attribute name="konyvRef" type="xs:string"/>
</xs:complexType>

<xs:complexType name="OsztalyfonokType">
    <xs:attribute name="ofID" type="xs:string"/>
    <xs:attribute name="tanarRef" type="xs:string"/>
    <xs:attribute name="osztalyRef" type="xs:string"/>
</xs:complexType>

<xs:complexType name="TanuloType">
    <xs:attribute name="osztalyRef" type="xs:string"/>
    <xs:attribute name="diakRef" type="xs:string"/>
</xs:complexType>

<xs:complexType name="KonyvType">
    <xs:attribute name="konyvID" type="xs:string"/>
    <xs:attribute name="TkonyvRef" type="xs:string"/>
    <xs:attribute name="targyRef" type="xs:string"/>
</xs:complexType>

<!-- Gyökér elem -->

<xs:element name="Iskola_ZF440N">
    <xs:complexType>
        <xs:sequence>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="Tanar" type="TanarType"/>
                <xs:element name="Tanit" type="TanitType"/>
                <xs:element name="Osztaly" type="OsztalyType"/>
                <xs:element name="Tantargy" type="TantargyType"/>
                <xs:element name="Tankonyv" type="TankonyvType"/>
                <xs:element name="Diak" type="DiakType"/>
                <xs:element name="Tanul" type="TanulType"/>
                <xs:element name="Osztalyfonok" type="OsztalyfonokType"/>
                <xs:element name="Tanulo" type="TanuloType"/>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
</xs:element>

```

```

        <xs:element name="Konyv" type="KonyvType"/>
    </xs:choice>
</xs:sequence>
</xs:complexType>
</xs:element>

</xs:schema>

```

Validáció sikeressége:

XML Validator – XSD (XML Schema)

Validators / XML Validator – XSD (XML Schema)

Validates the XML string/file against the specified XSD string/file. XSD files are "XML Schemas" that describe the structure of a XML document. The validator checks for well formedness first, meaning that your XML file must be parsable using a DOM/SAX parser, and only then does it validate your XML against the XML Schema. The validator will report fatal errors, non-fatal errors and warnings.

The XML document is valid.



(<https://www.freeformatter.com/xml-validator-xsd.html>)

2a) DOM file beolvasás

A kód egy XML fájlt vár bemenetként, amelyet feldolgoz a DOM parser segítségével. A program az XML fájlom adatstruktúráját kezeli. Az XML minden típusát kezeli, majd megjeleníti konzolon a beolvasott adatokat.

```

package hu.domparse.zf440n;

import java.io.File;
import java.io.IOException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DOMReadZF440N {

    private static final String FILE_NAME =
"E:\\ZF440N_XMLGyak\\XMLTaskZF440N\\XMLZF440N.xml";

    public static void main(String[] args) {
        try {
            Document document = parseXml(FILE_NAME);

```

```

        document.getDocumentElement().normalize();
        System.out.println("<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\n");
        System.out.println("<Iskola_ZF440N
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation=\"XMLSchemaZF440N.xsd\">\n");

        readTanar(document);
        printEmptyLine();

        readTanit(document);
        printEmptyLine();

        readOszталy(document);
        printEmptyLine();

        readTantargy(document);
        printEmptyLine();

        readTankonyv(document);
        printEmptyLine();

        readDiak(document);
        printEmptyLine();

        readTanul(document);
        printEmptyLine();

        System.out.println("\n</Iskola_ZF440N>");
    } catch (ParserConfigurationException | IOException | SAXException e)
    {
        handleException(e);
    }
}

private static void readTanar(Document document) {
    NodeList tanarList = document.getElementsByTagName("Tanar");
    for (int temp = 0; temp < tanarList.getLength(); temp++) {
        Node node = tanarList.item(temp);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            // Extract the necessary information from the new XML
            structure
            String vnev =
eElement.getElementsByTagName("vnev").item(0).getTextContent();
            String knev =
eElement.getElementsByTagName("knev").item(0).getTextContent();

```



```

        String irsz =
eElement.getElementsByTagName("irsz").item(0).getTextContent();
        String varos =
eElement.getElementsByTagName("varos").item(0).getTextContent();
        String utca =
eElement.getElementsByTagName("utca").item(0).getTextContent();
        String hsz =
eElement.getElementsByTagName("hsz").item(0).getTextContent();
        String ev =
eElement.getElementsByTagName("ev").item(0).getTextContent();
        String ho =
eElement.getElementsByTagName("ho").item(0).getTextContent();
        String nap =
eElement.getElementsByTagName("nap").item(0).getTextContent();
        String telefon =
eElement.getElementsByTagName("telefon").item(0).getTextContent();

        // Print the extracted information
        System.out.println("    <Tanar>");
        printElement("nev", vnev + " " + knev);
        System.out.println("        <lakcim>");
        printElement("irsz", irsz);
        printElement("varos", varos);
        printElement("utca", utca);
        printElement("hsz", hsz);
        System.out.println("        </lakcim>");
        System.out.println("        <szulido>");
        printElement("ev", ev);
        printElement("ho", ho);
        printElement("nap", nap);
        System.out.println("        </szulido>");
        printElement("telefon", telefon);
        System.out.println("    </Tanar>");
    }
}

private static void readTanit(Document document) {
    NodeList tanitList = document.getElementsByTagName("Tanit");
    for (int temp = 0; temp < tanitList.getLength(); temp++) {
        Node node = tanitList.item(temp);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            String szint =
eElement.getElementsByTagName("szint").item(0).getTextContent();
            String miota =
eElement.getElementsByTagName("miota").item(0).getTextContent();

            System.out.println("    <Tanit>");

```

```

        printElement("szint", szint);
        printElement("miota", miota);
        System.out.println("    </Tanit>");
    }
}

private static void readOsztaly(Document document) {
    NodeList osztalyList = document.getElementsByTagName("Osztaly");
    for (int temp = 0; temp < osztalyList.getLength(); temp++) {
        Node node = osztalyList.item(temp);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            String letszam =
eElement.getElementsByTagName("letszam").item(0).getTextContent();
            String tagozat =
eElement.getElementsByTagName("tagozat").item(0).getTextContent();

            System.out.println("    <Osztaly>");
            printElement("letszam", letszam);
            printElement("tagozat", tagozat);
            System.out.println("    </Osztaly>");
        }
    }
}

private static void readTantargy(Document document) {
    NodeList tantargyList = document.getElementsByTagName("Tantargy");
    for (int temp = 0; temp < tantargyList.getLength(); temp++) {
        Node node = tantargyList.item(temp);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            String tantargyNev =
eElement.getElementsByTagName("nev").item(0).getTextContent();

            System.out.println("    <Tantargy>");
            printElement("nev", tantargyNev);
            System.out.println("    </Tantargy>");
        }
    }
}

private static void readTankonyv(Document document) {
    NodeList tankonyvList = document.getElementsByTagName("Tankonyv");
    for (int temp = 0; temp < tankonyvList.getLength(); temp++) {
        Node node = tankonyvList.item(temp);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;

```

```

        String cim =
eElement.getElementsByTagName("cim").item(0).getTextContent();
        String iro =
eElement.getElementsByTagName("iro").item(0).getTextContent();
        String kiado =
eElement.getElementsByTagName("kiado").item(0).getTextContent();
        String kiadido =
eElement.getElementsByTagName("kiadido").item(0).getTextContent();

        System.out.println("    <Tankonyv>");
        printElement("cim", cim);
        printElement("iro", iro);
        printElement("kiado", kiado);
        printElement("kiadido", kiadido);
        System.out.println("    </Tankonyv>");
    }
}

private static void readDiak(Document document) {
    NodeList diakList = document.getElementsByTagName("Diak");
    for (int temp = 0; temp < diakList.getLength(); temp++) {
        Node node = diakList.item(temp);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            String vnev =
eElement.getElementsByTagName("vnev").item(0).getTextContent();
            String knev =
eElement.getElementsByTagName("knev").item(0).getTextContent();
            String irsz =
eElement.getElementsByTagName("irsz").item(0).getTextContent();
            String varos =
eElement.getElementsByTagName("varos").item(0).getTextContent();
            String utca =
eElement.getElementsByTagName("utca").item(0).getTextContent();
            String hsz =
eElement.getElementsByTagName("hsz").item(0).getTextContent();
            String ev =
eElement.getElementsByTagName("ev").item(0).getTextContent();
            String ho =
eElement.getElementsByTagName("ho").item(0).getTextContent();
            String nap =
eElement.getElementsByTagName("nap").item(0).getTextContent();
            String anyjaneve =
eElement.getElementsByTagName("anyjaneve").item(0).getTextContent();
            String kor =
eElement.getElementsByTagName("kor").item(0).getTextContent();

            System.out.println("    <Diak>");

```

```

        printElement("nev", vnev + " " + knev);
        System.out.println("        <lakcim>");
        printElement("irsz", irsz);
        printElement("varos", varos);
        printElement("utca", utca);
        printElement("hsz", hsz);
        System.out.println("        </lakcim>");
        System.out.println("        <szulido>");
        printElement("ev", ev);
        printElement("ho", ho);
        printElement("nap", nap);
        System.out.println("        </szulido>");
        printElement("anyjaneve", anyjaneve);
        printElement("kor", kor);
        System.out.println("    </Diak>");
    }
}

private static void readTanul(Document document) {
    NodeList tanullist = document.getElementsByTagName("Tanul");
    for (int temp = 0; temp < tanullist.getLength(); temp++) {
        Node node = tanullist.item(temp);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            String erdemjegy =
eElement.getElementsByTagName("erdemjegy").item(0).getTextContent();
            String datum =
eElement.getElementsByTagName("datum").item(0).getTextContent();

            System.out.println("    <Tanul>");
            printElement("erdemjegy", erdemjegy);
            printElement("datum", datum);
            System.out.println("    </Tanul>");
        }
    }
}

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

private static void printEmptyLine() {
    System.out.println();
}

private static Document parseXml(String fileName) throws
ParserConfigurationException, IOException, SAXException {

```

```
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        return builder.parse(new File(fileName));
    }

    private static void handleException(Exception e) {
        // Handle the exception (e.g., log it)
        e.printStackTrace();
    }
}
```

2b) DOM adatmódosítás

A program beolvassa az XMLZF440N.xml fájlt ezután végrehajt néhány módosítást, amit kiír a konzolra valamint a beolvasott fájlban is elvégzi a változtatásokat.

- Az első tanár telefonszámát megváltoztatja
- Az egyik diák életkorát 20-ra állítja
- Egy tantárgy szintje alapra állítódik
- Az egyik tankönyv új kiadót kap

```
package hu.domparse.zf440n;

import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.*;

public class DOMModifyZF440N {

    public static void main(String[] args) {
        try {
            File inputFile = new
File("E:\\ZF440N_XMLGyak\\XMLTaskZF440N\\XMLZF440N.xml");

            DocumentBuilderFactory docFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
            Document doc = docBuilder.parse(inputFile);

            // Módosítások végrehajtása
            modifyTelefon(doc, "1", "NewPhoneNumber");
            modifyKor(doc, "1", 20);
            modifySzint(doc, "1", "Alap");
            modifyKiado(doc, "1", "NewPublisher");

            // Visszaírás az XML fájlba
            writeDocumentToFile(doc,
"E:\\ZF440N_XMLGyak\\XMLTaskZF440N\\XMLZF440N.xml");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void modifyTelefon(Document doc, String tanarID, String
newPhoneNumber) {
        NodeList tanarList = doc.getElementsByTagName("Tanar");
        for (int i = 0; i < tanarList.getLength(); i++) {
```

```

        Node tanar = tanarList.item(i);
        Element tanarElement = (Element) tanar;

        if (tanarElement.getAttribute("tanarID").equals(tanarID)) {
            tanarElement.getElementsByTagName("telefon").item(0).setTextContent(newPhoneNumber);
        }
    }
}

private static void modifyKor(Document doc, String diakID, int newAge) {
    NodeList diakList = doc.getElementsByTagName("Diak");
    for (int i = 0; i < diakList.getLength(); i++) {
        Node diak = diakList.item(i);
        Element diakElement = (Element) diak;

        if (diakElement.getAttribute("diakID").equals(diakID)) {
            diakElement.getElementsByTagName("kor").item(0).setTextContent(Integer.toString(newAge));
        }
    }
}

private static void modifySzint(Document doc, String tanarID, String newSzint) {
    NodeList tanitList = doc.getElementsByTagName("Tanit");
    for (int i = 0; i < tanitList.getLength(); i++) {
        Node tanit = tanitList.item(i);
        Element tanitElement = (Element) tanit;

        if (tanitElement.getAttribute("tanarRef").equals(tanarID)) {
            tanitElement.getElementsByTagName("szint").item(0).setTextContent(newSzint);
        }
    }
}

private static void modifyKiado(Document doc, String konyvID, String newPublisher) {
    NodeList konyvList = doc.getElementsByTagName("Tankonyv");
    for (int i = 0; i < konyvList.getLength(); i++) {
        Node konyv = konyvList.item(i);
        Element konyvElement = (Element) konyv;

        if (konyvElement.getAttribute("konyvID").equals(konyvID)) {
            konyvElement.getElementsByTagName("kiado").item(0).setTextContent(newPublisher);
        }
    }
}

```

```
}

    private static void writeDocumentToFile(Document doc, String filename)
throws TransformerException {
    TransformerFactory transformerFactory =
TransformerFactory.newInstance();
    try {
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new File(filename));
        transformer.transform(source, result);
        System.out.println("The content has been written to the output
file successfully.");
    } catch (TransformerConfigurationException e) {
        e.printStackTrace();
    }
}
}
```


2c) DOM adat lekérdezés

A program a következő lekérdezéseket hajtja végre külön függvényekben.

- queryBPTanarok: a budapesti származású tanárokat keresi.
- queryDiakokTantargybol: azok a tanulók jelennek meg akik fizikát tanulnak.
- queryTanarokAdatai: az összes tanár neve és lakcíme kiíródik konzolra.
- queryOsztalyokLetszamTantargyankent: Az egyes tantárgyat látogató diákok száma.

```
package hu.domparse.zf440n;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

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

public class DOMQueryZF440N {

    public static void main(String[] args) {
        try {
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document document =
db.parse("E:\\ZF440N_XMLGyak\\XMLTaskZF440N\\XMLZF440N.xml");

            // Példa: Miskolci tanárok kilistázása
            queryBPTanarok(document);
            queryDiakokTantargybol(document, "Fizika");
            queryTanarokAdatai(document);
            queryOsztalyokLetszamTantargyankent(document);

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

    // Budapesti tanárok kilistázása
    private static void queryBPTanarok(Document document) {
        System.out.println("=== Budapesti tanárok ===");

        NodeList tanarList = document.getElementsByTagName("Tanar");

        for (int i = 0; i < tanarList.getLength(); i++) {
            Node tanarNode = tanarList.item(i);

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

```

        Element tanarElement = (Element) tanarNode;

        String varos =
tanarElement.getElementsByTagName("varos").item(0).getTextContent();

        if (varos.equalsIgnoreCase("Budapest")) {
            // Ha a tanár Miskolcon lakik, kiírjuk az adatait
            String nev =
tanarElement.getElementsByTagName("nev").item(0).getTextContent();
            String szuletesiEv =
tanarElement.getElementsByTagName("ev").item(0).getTextContent();

            System.out.println("    <Tanar>");
            System.out.println("        <Nev>" + nev + "</Nev>");
            System.out.println("        <SzuletesiEv>" + szuletesiEv +
"</SzuletesiEv>");
            System.out.println("    </Tanar>");
            System.out.println("\n");
        }
    }

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

private static void queryDiakokTantargybol(Document document, String
tantargyNev) {
    System.out.println("=== Diákok a(z) " + tantargyNev + " tantárgyból
===");

    NodeList tanullist = document.getElementsByTagName("Tanul");
    NodeList tantargylist = document.getElementsByTagName("Tantargy");

    for (int i = 0; i < tantargylist.getLength(); i++) {
        Node tantargynode = tantargylist.item(i);
        Element tantargyElement = (Element) tantargynode;

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

        if (nev.equalsIgnoreCase(tantargyNev)) {
            // Az adott tantárgyhoz tartozó diákok kilistázása
            for (int j = 0; j < tanullist.getLength(); j++) {
                Node tanulNode = tanullist.item(j);

                if (tanulNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element tanulElement = (Element) tanulNode;

```

```

        String erdemjegy =
tanulElement.getElementsByTagName("erdemjegy").item(0).getTextContent();
        String datum =
tanulElement.getElementsByTagName("datum").item(0).getTextContent();
        String diakNev = getDiakNevById(document,
tanulElement.getAttribute("diakRef"));

        System.out.println("    <Diak>");
        System.out.println("        <Nev>" + diakNev + "</Nev>");
        System.out.println("        <Erdemjegy>" + erdemjegy +
"</Erdemjegy>");
        System.out.println("        <Datum>" + datum +
"</Datum>");
        System.out.println("    </Diak>");
        System.out.println("\n");
    }
}
}

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

private static String getDiakNevById(Document document, String diakId) {
    NodeList diakList = document.getElementsByTagName("Diak");

    for (int i = 0; i < diakList.getLength(); i++) {
        Node diakNode = diakList.item(i);

        if (diakNode.getNodeType() == Node.ELEMENT_NODE) {
            Element diakElement = (Element) diakNode;

            String currentDiakId = diakElement.getAttribute("diakID");

            if (currentDiakId.equals(diakId)) {
                // Az adott diák nevének összeállítása
                String vnev =
diakElement.getElementsByTagName("vnev").item(0).getTextContent();
                String knev =
diakElement.getElementsByTagName("knev").item(0).getTextContent();

                return vnev + " " + knev;
            }
        }
    }

    return "";
}
}

```

```

        private static void queryTanarokAdatai(Document document) {
            System.out.println("=== Tanárok születési évének és lakhelyének megjelenítése ===");

            NodeList tanarList = document.getElementsByTagName("Tanar");

            for (int i = 0; i < tanarList.getLength(); i++) {
                Node tanarNode = tanarList.item(i);

                if (tanarNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element tanarElement = (Element) tanarNode;

                    String nev =
tanarElement.getElementsByTagName("nev").item(0).getTextContent();
                    String szuletesiEv =
tanarElement.getElementsByTagName("ev").item(0).getTextContent();
                    String varos =
tanarElement.getElementsByTagName("varos").item(0).getTextContent();

                    System.out.println("    <Tanar>");
                    System.out.println("        <Nev>" + nev + "</Nev>");
                    System.out.println("        <SzuletesiEv>" + szuletesiEv +
"</SzuletesiEv>");
                    System.out.println("        <Lakohely>" + varos + "</Lakohely>");
                    System.out.println("    </Tanar>");
                    System.out.println("\n");
                }
            }

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

        private static void queryOsztalyokLetszamTantargyankent(Document document)
        {
            System.out.println("=== Osztályok létszámainak összegzése tantárgyanként ===");

            NodeList osztalyList = document.getElementsByTagName("Osztaly");
            NodeList tanuloList = document.getElementsByTagName("Tanulo");

            for (int i = 0; i < osztalyList.getLength(); i++) {
                Node osztalyNode = osztalyList.item(i);
                Element osztalyElement = (Element) osztalyNode;

                String tantargy =
osztalyElement.getElementsByTagName("tagozat").item(0).getTextContent();
                int osztalyLetszam =
Integer.parseInt(osztalyElement.getElementsByTagName("letszam").item(0).getTex
tContent());

```

```

        int tantargyLetszam = 0;

        for (int j = 0; j < tanuloList.getLength(); j++) {
            Node tanuloNode = tanuloList.item(j);

            if (tanuloNode.getNodeType() == Node.ELEMENT_NODE) {
                Element tanuloElement = (Element) tanuloNode;

                String osztalyRef =
tanuloElement.getAttribute("osztalyRef");

                if
(osztalyRef.equals(osztalyElement.getAttribute("osztalyID"))) {
                    tantargyLetszam++;
                }
            }
        }

        System.out.println("    <Osztaly>");
        System.out.println("        <Tantargy>" + tantargy + "</Tantargy>");
        System.out.println("        <Letszam>" + tantargyLetszam +
"</Letszam>");
        System.out.println("    </Osztaly>");
        System.out.println("\n");
    }

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

```

2d) DOM adatírás

Ez a program egy DOM (Document Object Model) alapú XML fájlírást valósít meg.

Néhány fontos jellemzője a következő:

Fájlműveletek kezelése: A program beolvassa az XML fájlt és DOM dokumentumot hoz létre.

DOM Fa bejárása: A printNode metódus segítségével a program rekurzívan bejárja a DOM fát, és kiírja az XML elemeket és attribútumokat. Az indentálás révén jól láthatóvá teszi a fa struktúráját.

Dokumentum írása fájlba: A writeDocumentToFile metódus segítségével a program kimentti a feldolgozott DOM dokumentumot egy új XML fájlba (XMLZF440N1.xml). Az írás során beállítja az indentálást is.

Kimeneti üzenet: A program kiírja a konzolra az "The content has been written to the output file successfully." üzenetet, ha sikeresen végrehajtotta a fájlírást.

Hibakezelés: A kódban található try-catch blokk segítségével kezeli a kivételeket és kiírja a hibaüzeneteket a hibakeresés megkönnyítése érdekében.

```
package hu.domparse.zf440n;

import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.*;

public class DOMWriteZF440N {

    public static void main(String[] args) {
        try {
            File inputFile = new
File("E:\\ZF440N_XMLGyak\\XMLTaskZF440N\\XMLZF440N.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            printNode(doc.getDocumentElement(), "");
            writeDocumentToFile(doc, "XMLZF440N1.xml");

            System.out.println("The content has been written to the output
file successfully.");
        } catch (Exception e) {
```

```

        e.printStackTrace();
    }
}

private static void printNode(Node node, String indent) {
    if (node.getNodeType() == Node.ELEMENT_NODE) {
        System.out.print("\n" + indent + "<" + node.getNodeName());
        if (node.hasAttributes()) {
            NamedNodeMap nodeMap = node.getAttributes();
            for (int i = 0; i < nodeMap.getLength(); i++) {
                Node attr = nodeMap.item(i);
                System.out.print(" " + attr.getNodeName() + "=\"" +
attr.getNodeValue() + "\"");
            }
        }

        NodeList children = node.getChildNodes();
        if (children.getLength() == 1 && children.item(0).getNodeType() ==
Node.TEXT_NODE) {
            System.out.print(">" +
children.item(0).getTextContent().trim());
            System.out.println("</" + node.getNodeName() + ">");
        } else {
            System.out.println(">");
            for (int i = 0; i < children.getLength(); i++)
                printNode(children.item(i), indent + "    ");
            System.out.println(indent + "</" + node.getNodeName() + ">");
        }
    } else if (node.getNodeType() == Node.TEXT_NODE) {
        String content = node.getTextContent().trim();
        if (!content.isEmpty())
            System.out.print(content);
    }
}

private static void writeDocumentToFile(Document doc, String filename)
throws TransformerException {
    TransformerFactory transformerFactory =
TransformerFactory.newInstance();
    try {
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new File(filename));
        transformer.transform(source, result);
    } catch (TransformerConfigurationException e) {
        e.printStackTrace();
    }
}
}

```

}