JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Szobafoglalás

Készítette: **Zavarkó Máté**

Neptunkód: **IN3BLK**

Dátum: **2023.12.04**

Tartalomjegyzék:

Beadandó témája....3
1. Feladat....3
1a) Az adatbázis ER modell tervezése....3
1b) Az adatbázis konvertálása XDM modellre....5
1c) Az XDM modell alapján XML dokumentum készítése....5
1d) Az XML dokumentum alapján XMLSchema készítése....9
2. Feladat....14
2a) adatolvasás....14
2b) adatmódosítás....17
2c) adatlekérdezés....19
2d) adatírás....25

Beadandó témája

A beadandó témája egy olyan adatbázis, amely különböző hotelek szobafoglalásait kezeli. Megmutatja, hogy a személyek milyen és mennyi szobát foglaltak le az adott hotelben. Lekérdezhetjük a vevő adatait, ami alapján a számla is készül. Továbbá megtalálható a hotelek adatai és az értékeléseik.

1. Feladat

1a) Az adatbázis ER modell tervezése

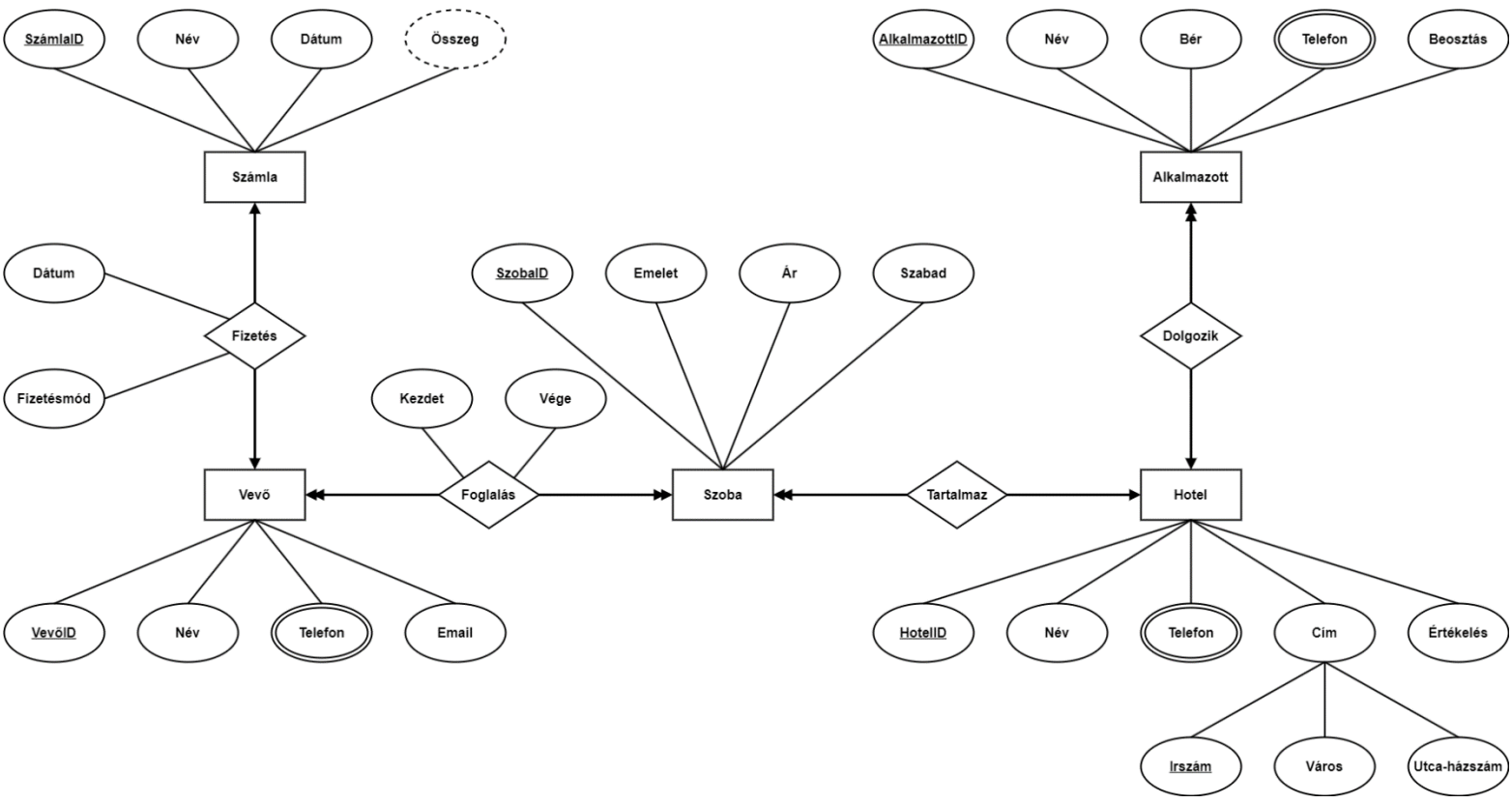
- **Vevő:**
  - *VevőID:* vevő elsődleges kulcsa
  - *Név:* vevő neve
  - *Telefon:* vevő telefonszáma, többértékű tulajdonság
  - *Email:* vevő email címe
- **Számla:**
  - *SzámlaID:* számla elsődleges kulcsa
  - *Név:* vevő neve, akihez tartozik a számla
  - *Dátum:* számla előállításának ideje
  - *Összeg:* fizetendő összeg
- **Szoba:**
  - *SzobaID:* szoba elsődleges kulcsa
  - *Emelet:* melyik emeleten található a szoba
  - *Típus:* szoba ágyainak mennyisége
  - *Ár:* az adott szoba ára egy éjszakára
  - *Szabad:* szoba foglaltsága
- **Hotel:**
  - *HotelID:* hotel elsődleges kulcsa
  - *Név:* hotel neve
  - *Telefon:* hotel telefonszáma, többértékű tulajdonság
  - *Cím:* hotel címe, összetett tulajdonság
  - *Értékelés:* hotel értékelése
- **Alkalmazott:**
  - *AlkalmazottID:* alkalmazott elsődleges kulcsa
  - *Név:* alkalmazott neve
  - *Bér:* alkalmazott bére
  - *Telefon:* alkalmazott telefonszáma
  - *Beosztás:* alkalmazott beosztása

Kapcsolatok:

- **Vevő és Számla (Fizetés)**
  Vevő és a *Számla* között egy-egy (1:1) kapcsolat van, mivel egy vevőhöz egy számla, illetve egy számla egy vevőhöz tartozik. Hozzátartozói a *Dátum*, hogy mikor történt a fizetés, és a *Fizetésmód*, hogy hogyan fizetett a vevő.
- **Vevő és Szoba (Foglalás)**
  Vevő és a *Szoba* között több-több (N:M) kapcsolat van, hiszen egy vevő kivehet több szobát és egy szobát több ember is kivehet. Hozzátartozói a *Kezde*t és a *Vége,* amik mutatják, hogy mettől meddig foglalta le a vevő a szobát.
- **Szoba és Hotel (Tartalmaz)**
  Szoba és *Hotel* között egy-több (1:N) kapcsolat van, ugyanis egy szoba egy hotelhez tartozik, de egy hotelhez több szoba is tartozik.
- **Hotel és Alkalmazott (Dolgozik)**

  Hotel és *Alkalmazott* egy-több (1:N) kapcsolat van, mivel egy hotelnek lehet több alkalmazottja, viszont egy alkalmazott egy hotelben dolgozhat.

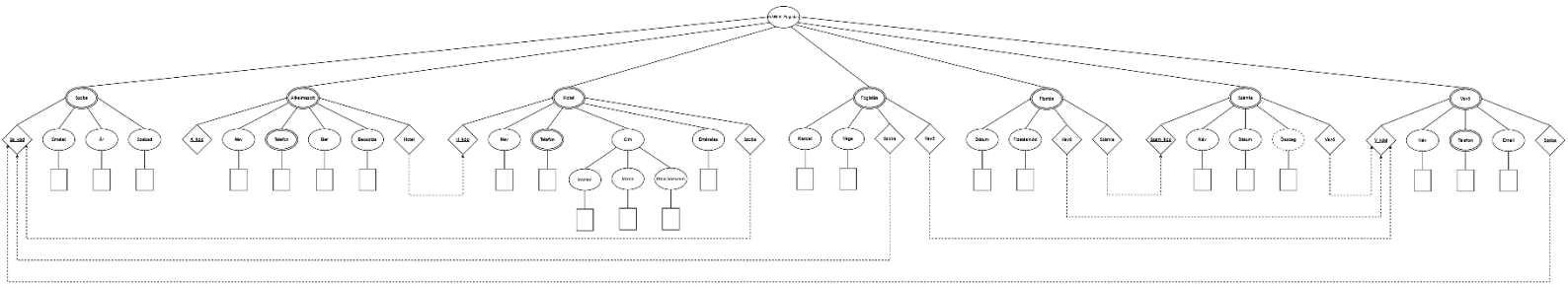
ER Modell:



1b) Az adatbázis konvertálása XDM modellre

Az XDM modell használatakor háromféle jelölést alkalmazhatunk. Az elemeket ellipszis ábrázolja, minden egyedből és a tulajdonságokból elem lesz. Az attribútumokat rombusz jelöli, melyek a kulcs tulajdonságokból erednek. A szöveget, amely az XML dokumentumban megjelenik, téglalap ábrázolja. Azok az elemek, amelyek többször is előfordulhatnak, dupla ellipszissel vannak jelölve. Az idegenkulcsok és a kulcsok közötti kapcsolatot szaggatott vonalas nyíl jelzi.

XDM Modell:



1c) Az XDM modell alapján XML dokumentum készítése

Az XML dokumentumot az XDM modell alapján elkészítettem, kezdve a root (gyökér) elemmel, amely az " IN3BLK_Foglalas ". Létrehoztam 3-3 példányt a gyermek elemekből, melyek attribútumai tartalmazzák a kulcsokat és idegenkulcsokat is. Ezt követően ezekhez az elemekhez létrehoztam a további gyermek elemeket is.

XML forráskód:

```xml
<?xml version="1.0" encoding="utf-8"?>
<IN3BLK_Foglalas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="XMLSchemaIN3BLK.xsd">

    <!-- Szobák -->
    <Szoba sz_kod="1">
        <Emelet>2</Emelet>
        <Ar>36000</Ar>
        <Szabad>Szabad</Szabad>
    </Szoba>

    <Szoba sz_kod="2">
        <Emelet>5</Emelet>
        <Ar>43000</Ar>
        <Szabad>Szabad</Szabad>
    </Szoba>
```

```xml
<Szoba sz_kod="3">
    <Emelet>1</Emelet>
    <Ar>31000</Ar>
    <Szabad>Foglalt</Szabad>
</Szoba>

<!-- Alkalmazottak -->
<Alkalmazott a_kod="1" h_kod="1">
    <Nev>Tóth András</Nev>
    <Telefon>06-70-938-5617</Telefon>
    <Ber>320000</Ber>
    <Beosztas>Portás</Beosztas>
</Alkalmazott>

<Alkalmazott a_kod="2" h_kod="2">
    <Nev>Lakatos Kevin</Nev>
    <Telefon>06-30-294-7335</Telefon>
    <Telefon>06-20-993-3744</Telefon>
    <Ber>390000</Ber>
    <Beosztas>Szakács</Beosztas>
</Alkalmazott>

<Alkalmazott a_kod="3" h_kod="3">
    <Nev>Szabó Ilona</Nev>
    <Telefon>06-70-223-9485</Telefon>
    <Ber>280000</Ber>
    <Beosztas>Takarító</Beosztas>
</Alkalmazott>

<!-- Hotelek -->
<Hotel h_kod="1" sz_kod="1">
    <Nev>Kényelem Hotel</Nev>
    <Telefon>06-20-993-7766</Telefon>
    <Cim>
        <Irszam>1028</Irszam>
        <Varos>Budapest</Varos>
        <Utca-hazszam>Szeles utca 25</Utca-hazszam>
    </Cim>
    <Ertekeles>3</Ertekeles>
</Hotel>

<Hotel h_kod="2" sz_kod="2">
    <Nev>Royal Hotel</Nev>
    <Telefon>06-70-234-3443</Telefon>
    <Cim>
        <Irszam>4033</Irszam>
        <Varos>Debrecen</Varos>
        <Utca-hazszam>Erdei utca 6</Utca-hazszam>
```

```
        </Cim>
        <Ertekeles>5</Ertekeles>
    </Hotel>

    <Hotel h_kod="3" sz_kod="3">
        <Nev>Udvari Hotel</Nev>
        <Telefon>06-30-345-2345</Telefon>
        <Telefon>06-30-999-2384</Telefon>
        <Cim>
            <Irszam>3521</Irszam>
            <Varos>Miskolc</Varos>
            <Utca-hazszam>Jakab utca 56</Utca-hazszam>
        </Cim>
        <Ertekeles>4</Ertekeles>
    </Hotel>

    <!-- Számlák -->
    <Szamla szam_kod="1" v_kod="1">
        <Nev>Kiss Anna</Nev>
        <Datum>2020-02-17</Datum>
        <Osszeg>86000</Osszeg>
    </Szamla>

    <Szamla szam_kod="2" v_kod="2">
        <Nev>Török András</Nev>
        <Datum>2021-09-10</Datum>
        <Osszeg>113000</Osszeg>
    </Szamla>

    <Szamla szam_kod="3" v_kod="3">
        <Nev>Horváth Áron</Nev>
        <Datum>2022-05-14</Datum>
        <Osszeg>93000</Osszeg>
    </Szamla>

    <!-- Vevők -->
    <Vevo v_kod="1" sz_kod="1">
        <Nev>Kiss Anna</Nev>
        <Telefon>06-30-222-2345</Telefon>
        <Telefon>06-70-399-5577</Telefon>
        <Email>kAnna@gmail.com</Email>
    </Vevo>

    <Vevo v_kod="2" sz_kod="2">
        <Nev>Török András</Nev>
        <Telefon>06-20-948-3857</Telefon>
        <Telefon>06-30-947-5872</Telefon>
        <Email>tAndras@freemail.hu</Email>
    </Vevo>
```

```
    <Vevo v_kod="3" sz_kod="3">
        <Nev>Horváth Áron</Nev>
        <Telefon>06-70-993-6665</Telefon>
        <Email>hAron@gmail.com</Email>
    </Vevo>

    <!-- Foglalások -->
    <Foglalas sz_kod="1" v_kod="1">
        <Kezdet>2020-02-10</Kezdet>
        <Vege>2020-02-17</Vege>
    </Foglalas>

    <Foglalas sz_kod="2" v_kod="2">
        <Kezdet>2021-09-03</Kezdet>
        <Vege>2021-09-10</Vege>
    </Foglalas>

    <Foglalas sz_kod="3" v_kod="3">
        <Kezdet>2022-05-10</Kezdet>
        <Vege>2022-05-14</Vege>
    </Foglalas>

    <!-- Fizetések -->
    <Fizetes v_kod="1" szam_kod="1">
        <Datum>2020-02-17</Datum>
        <Fizetesmod>Készpénz</Fizetesmod>
    </Fizetes>

    <Fizetes v_kod="2" szam_kod="2">
        <Datum>2021-09-10</Datum>
        <Fizetesmod>Bankkártya</Fizetesmod>
    </Fizetes>

    <Fizetes v_kod="3" szam_kod="3">
        <Datum>2022-05-14</Datum>
        <Fizetesmod>Bankkártya</Fizetesmod>
    </Fizetes>
</IN3BLK_Foglalas>
```

1d) Az XML dokumentum alapján XMLSchema készítése

Ezután elkészítem az XML-ben meghatározott típusokat és kulcsokat megszabó sémát, kigyűjtöm az egyszerű típusokat, elemeket, és ezek megszabásait beállítom. Ezt követően definiálom a saját, komplex típusaimat, ezekre is alkalmazom a megszabásokat, az egyszerű típusok felhasználásával. Ezen lépések után a gyökérelemtől kiindulva felépítem az XML struktúrát, meghatározom az elsődleges kulcsokat, valamint ezekhez az elsődleges kulcsokhoz tartozó idegenkulcsokat. Emellett az 1:1 kapcsolat megvalósításához használom a "Unique"-t is.

XML forráskód:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
    <!-- Saját típusok, egyszerű típusok -->
    <xs:element name="Emelet" type="xs:integer" />
    <xs:element name="Ar" type="xs:integer" />
    <xs:element name="Szabad" type="xs:string" />
    <xs:element name="Nev" type="xs:string" />
    <xs:element name="Telefon" type="telefonszamTipus" />
    <xs:element name="Ber" type="xs:integer" />
    <xs:element name="Beosztas" type="xs:string" />
    <xs:element name="Cim" type="xs:string" />
    <xs:element name="Irszam" type="iranyitoszamTipus" />
    <xs:element name="Varos" type="xs:string" />
    <xs:element name="Utca-hazszam" type="xs:string" />
    <xs:element name="Ertekeles" type="xs:integer" />
    <xs:element name="Kezdet" type="datumTipus" />
    <xs:element name="Vege" type="datumTipus" />
    <xs:element name="Datum" type="datumTipus" />
    <xs:element name="Fizetesmod" type="xs:string" />
    <xs:element name="Osszeg" type="xs:integer" />
    <xs:element name="Email" type="xs:string" />

    <xs:simpleType name="telefonszamTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="\d{2}-\d{2}-\d{3}-\d{4}" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="iranyitoszamTipus">
        <xs:restriction base="xs:integer">
            <xs:pattern value="\d{4}" />
        </xs:restriction>
    </xs:simpleType>
```

```
    <xs:simpleType name="datumTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="\d{4}-\d{2}-\d{2}" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="szabadTipus">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Szabad" />
            <xs:enumeration value="Foglalt" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="ertekelesTipus">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1" />
            <xs:maxInclusive value="5" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Komplex típusok -->
    <xs:complexType name="szobaTipus">
        <xs:sequence>
            <xs:element ref="Emelet" />
            <xs:element ref="Ar" />
            <xs:element ref="Szabad" />
        </xs:sequence>
        <xs:attribute name="sz_kod" type="xs:integer" />
    </xs:complexType>

    <xs:complexType name="alkalmazottTipus">
        <xs:sequence>
            <xs:element ref="Nev" />
            <xs:element ref="Telefon" maxOccurs="unbounded" />
            <xs:element ref="Ber" />
            <xs:element ref="Beosztas" />
        </xs:sequence>
        <xs:attribute name="a_kod" type="xs:integer" />
        <xs:attribute name="h_kod" type="xs:integer" />
    </xs:complexType>

    <xs:complexType name="hotelTipus">
        <xs:sequence>
            <xs:element ref="Nev" />
            <xs:element ref="Telefon" maxOccurs="unbounded" />
            <xs:element name="Cim">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="Irszam" />
```

```
                    <xs:element ref="Varos" />
                    <xs:element ref="Utca-hazszam" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element ref="Ertekeles" />
    </xs:sequence>
    <xs:attribute name="h_kod" type="xs:integer" />
    <xs:attribute name="sz_kod" type="xs:integer" />
</xs:complexType>

<xs:complexType name="szamlaTipus">
    <xs:sequence>
        <xs:element ref="Nev" />
        <xs:element ref="Datum" />
        <xs:element ref="Osszeg" />
    </xs:sequence>
    <xs:attribute name="szam_kod" type="xs:integer" />
    <xs:attribute name="v_kod" type="xs:integer" />
</xs:complexType>

<xs:complexType name="vevoTipus">
    <xs:sequence>
        <xs:element ref="Nev" />
        <xs:element ref="Telefon" maxOccurs="unbounded" />
        <xs:element ref="Email" />
    </xs:sequence>
    <xs:attribute name="v_kod" type="xs:integer" />
    <xs:attribute name="sz_kod" type="xs:integer" />
</xs:complexType>

<xs:complexType name="foglalasTipus">
    <xs:sequence>
        <xs:element ref="Kezdet" />
        <xs:element ref="Vege" />
    </xs:sequence>
    <xs:attribute name="sz_kod" type="xs:integer" />
    <xs:attribute name="v_kod" type="xs:integer" />
</xs:complexType>

<xs:complexType name="fizetesTipus">
    <xs:sequence>
        <xs:element ref="Datum" />
        <xs:element ref="Fizetesmod" />
    </xs:sequence>
    <xs:attribute name="v_kod" type="xs:integer" />
    <xs:attribute name="szam_kod" type="xs:integer" />
</xs:complexType>
```

```
    <!-- Gyökérelem -->
    <xs:element name="IN3BLK_Foglalas">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Szoba" type="szobaTipus" minOccurs="0"
maxOccurs="unbounded" />
                <xs:element name="Alkalmazott" type="alkalmazottTipus"
minOccurs="0" maxOccurs="unbounded" />
                <xs:element name="Hotel" type="hotelTipus" minOccurs="0"
maxOccurs="unbounded" />
                <xs:element name="Szamla" type="szamlaTipus" minOccurs="0"
maxOccurs="unbounded" />
                <xs:element name="Vevo" type="vevoTipus" minOccurs="0"
maxOccurs="unbounded" />
                <xs:element name="Foglalas" type="foglalasTipus" minOccurs="0"
maxOccurs="unbounded" />
                <xs:element name="Fizetes" type="fizetesTipus" minOccurs="0"
maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>

        <!-- Elsődleges kulcsok -->
        <xs:key name="HotelPK">
            <xs:selector xpath="Hotel" />
            <xs:field xpath="@h_kod" />
        </xs:key>

        <xs:key name="SzobaPK">
            <xs:selector xpath="Szoba" />
            <xs:field xpath="@sz_kod" />
        </xs:key>

        <xs:key name="AlkalmazottPK">
            <xs:selector xpath="Alkalmazott" />
            <xs:field xpath="@a_kod" />
        </xs:key>

        <xs:key name="SzamlaPK">
            <xs:selector xpath="Szamla" />
            <xs:field xpath="@szam_kod" />
        </xs:key>

        <xs:key name="VevoPK">
            <xs:selector xpath="Vevo" />
            <xs:field xpath="@v_kod" />
        </xs:key>

        <!-- Idegen kulcsok -->
        <xs:keyref name="HotelFK" refer="SzobaPK">
```

```
            <xs:selector xpath="Hotel" />
            <xs:field xpath="@sz_kod" />
        </xs:keyref>

        <xs:keyref name="AlkalmazottFK" refer="HotelPK">
            <xs:selector xpath="Alkalmazott" />
            <xs:field xpath="@h_kod" />
        </xs:keyref>

        <xs:keyref name="SzamlaFK" refer="VevoPK">
            <xs:selector xpath="Szamla" />
            <xs:field xpath="@v_kod" />
        </xs:keyref>

        <xs:keyref name="VevoFK" refer="SzobaPK">
            <xs:selector xpath="Vevo" />
            <xs:field xpath="@sz_kod" />
        </xs:keyref>

        <xs:keyref name="FoglalasSZFK" refer="SzobaPK">
            <xs:selector xpath="Foglalas" />
            <xs:field xpath="@sz_kod" />
        </xs:keyref>

        <xs:keyref name="FoglalasVFK" refer="VevoPK">
            <xs:selector xpath="Foglalas" />
            <xs:field xpath="@v_kod" />
        </xs:keyref>

        <xs:keyref name="FizetesSZFK" refer="SzamlaPK">
            <xs:selector xpath="Fizetes" />
            <xs:field xpath="@szam_kod" />
        </xs:keyref>

        <xs:keyref name="FizetesVFK" refer="VevoPK">
            <xs:selector xpath="Fizetes" />
            <xs:field xpath="@v_kod" />
        </xs:keyref>

        <!-- 1:1 kapcsolat-->
        <xs:unique name="Vevo_Szamla_kapcsolat">
            <xs:selector xpath="Vevo" />
            <xs:field xpath="@szam_kod" />
        </xs:unique>

    </xs:element>
</xs:schema>
```

2. Feladat

2a) adatolvasás

A három könyvtárak importálása után a fájl beolvasását végzem egy try-catch blokkban. Példányosítom a DocumentBuilderFactory-t, majd normalizálom a dokumentumot. Ezt követően megnyitom az output fájlt, és meghívom a printWriter nevű függvényt, amely egyszerre írja a bemeneti XML dokumentum tartalmát a konzolra és a fájlba. A dokumentum főbb elemeit NodeListekben tárolom el, majd ezeken egy for ciklus segítségével iterálok végig, és vizsgálom a gyerekelemek tartalmát.

```
package hu.domparse.IN3BLK;

import org.w3c.dom.*;

import javax.xml.parsers.*;
import java.io.*;
import java.util.StringJoiner;

public class DomReadIN3BLK {

    public static void main(String[] args) {
        try {
            // XML fájl beolvasása
            File inputFile = new
File("XMLTaskIN3BLK\\DOMParseIN3BLK\\XMLIN3BLK.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            // Mentés fájlba
            File outputFile = new
File("XMLTaskIN3BLK\\DOMParseIN3BLK\\ReadOutput.xml");
            PrintWriter writer = new PrintWriter(new FileWriter(outputFile,
true));

            // XML gyökér elemének kiíratása a konzolra és fájlba
            Element rootElement = doc.getDocumentElement();
            String rootName = rootElement.getTagName();
            StringJoiner rootAttributes = new StringJoiner(" ");
            NamedNodeMap rootAttributeMap = rootElement.getAttributes();

            for (int i = 0; i < rootAttributeMap.getLength(); i++) {
                Node attribute = rootAttributeMap.item(i);
```

```
                rootAttributes.add(attribute.getNodeName() + "=\"" +
attribute.getNodeValue() + "\"");
            }

            System.out.print("<?xml version=\"1.0\" encoding=\"UTF-8\"?> \n");
            writer.print("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");

            System.out.print("<" + rootName + " " + rootAttributes.toString()
+ "> \n");
            writer.print("<" + rootName + " " + rootAttributes.toString() + ">
\n");

            NodeList szobaList = doc.getElementsByTagName("Szoba");
            NodeList alkalmazottList =
doc.getElementsByTagName("Alkalmazott");
            NodeList hotelList = doc.getElementsByTagName("Hotel");
            NodeList szamlaList = doc.getElementsByTagName("Szamla");
            NodeList vevoList = doc.getElementsByTagName("Vevo");
            NodeList foglalasList = doc.getElementsByTagName("Foglalas");
            NodeList fizetesList = doc.getElementsByTagName("Fizetes");

            // XML kiírása az eredeti formában
            System.out.println("");
            writer.println("");
            printNodeList(szobaList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(alkalmazottList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(hotelList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(szamlaList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(vevoList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(foglalasList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(fizetesList, writer);

            // XML gyökér elemének lezárása
            System.out.println("</" + rootName + ">");
            writer.append("</" + rootName + ">");

            writer.close();
```

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

    }

    // NodeList kiírása
    private static void printNodeList(NodeList nodeList, PrintWriter writer) {
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            printNode(node, 1, writer);
            System.out.println("");
            writer.println("");
        }
    }

    // Node kiírása
    private static void printNode(Node node, int indent, PrintWriter writer) {
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;
            String nodeName = element.getTagName();
            StringJoiner attributes = new StringJoiner(" ");
            NamedNodeMap attributeMap = element.getAttributes();

            for (int i = 0; i < attributeMap.getLength(); i++) {
                Node attribute = attributeMap.item(i);
                attributes.add(attribute.getNodeName() + "=\"" +
attribute.getNodeValue() + "\"");
            }

            System.out.print(getIndentString(indent));
            System.out.print("<" + nodeName + " " + attributes.toString() +
">");
            writer.print(getIndentString(indent));
            writer.print("<" + nodeName + " " + attributes.toString() + ">");

            NodeList children = element.getChildNodes();
            // Ellenőrzi, hogy az elemnek csak egy szöveges tartalma van-e
            if (children.getLength() == 1 && children.item(0).getNodeType() ==
Node.TEXT_NODE) {
                // Ha csak egy szöveges tartalom van, akkor kiíratja
                System.out.print(children.item(0).getNodeValue());
                writer.print(children.item(0).getNodeValue());
            } else {
                // Ha több gyerek eleme van, akkor új sor karaktereket és
behúzást ad hozzá
                System.out.println();
                writer.println();
                for (int i = 0; i < children.getLength(); i++) {
```

```
                    printNode(children.item(i), indent + 1, writer);
                }
                System.out.print(getIndentString(indent));
                writer.print(getIndentString(indent));
            }
            System.out.println("</" + nodeName + ">");
            writer.println("</" + nodeName + ">");
        }

    }

    // Behúzások hozzáadása
    private static String getIndentString(int indent) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < indent; i++) {
            // Minden iteráció során két szóközt fűz hozzá a StringBuilderhez
            sb.append("  ");
        }
        return sb.toString();
    }

}
```

2b) adatmódosítás

A három könyvtárak importálása után a fájl beolvasását végzem egy try-catch blokkban. Példányosítom a DocumentBuilderFactory-t, majd normalizálom a dokumentumot. A dokumentum főbb elemeit NodeListekben tárolom el. Az elemek módosítását úgy végzem el, hogy lekérem az elem tartalmát a getElementsByTagName DOM függvénnyel. Ezt követően a setTextContent metódussal módosítom a tartalmát.

```
package hu.domparse.IN3BLK;

import javax.xml.parsers.*;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.*;
import java.io.*;

public class DomModifyIN3BLK {

    public static void main(String[] args) {
        try {
```

```
            // XML fájl beolvasása
            File inputFile = new
File("XMLTaskIN3BLK\\DOMParseIN3BLK\\XMLIN3BLK.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            // Adatok módosítása
            // Alkalmazott bérének módosítása
            // Egy adott típus összes elemét lekérjük és eltároljuk egy
listába
            NodeList alkalmazottList =
doc.getElementsByTagName("Alkalmazott");
            // Index alapján lekérjük azt az elemet, amelyiket módosítani
szeretnénk
            Element alkalmazott = (Element) alkalmazottList.item(0);
            // Megkeressük az elemben a módosítani kívánt tag-et, majd
beállítjuk az új
            // tartalmát (setTextContent)
            alkalmazott.getElementsByTagName("Ber").item(0).setTextContent("40
0000");

            // Hotel telefonszámának módosítása
            NodeList hotelList = doc.getElementsByTagName("Hotel");
            Element hotel = (Element) hotelList.item(1);
            hotel.getElementsByTagName("Telefon").item(0).setTextContent("06-
20-334-4977");

            // Vevő Email címének módosítása
            NodeList vevoList = doc.getElementsByTagName("Vevo");
            Element vevo = (Element) vevoList.item(2);
            vevo.getElementsByTagName("Email").item(0).setTextContent("horvat.
aron@gmail.com");

            // Foglalás végének időpontjának módosítása
            NodeList foglalasList = doc.getElementsByTagName("Foglalas");
            Element foglalas = (Element) foglalasList.item(0);
            foglalas.getElementsByTagName("Vege").item(0).setTextContent("2020
-02-23");

            // Fizetés módjának módosítása
            NodeList fizetesList = doc.getElementsByTagName("Fizetes");
            Element fizetes = (Element) fizetesList.item(2);
            fizetes.getElementsByTagName("Fizetesmod").item(0).setTextContent(
"Készpénz");

            // Konzolra való kiíratás
```

```
            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            DOMSource source = new DOMSource(doc);
            StreamResult consoleResult = new StreamResult(System.out);
            transformer.transform(source, consoleResult);

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

    }

}
```

2c) adatlekérdezés

A három könyvtárak importálása után a fájl beolvasását végzem egy try-catch blokkban. Példányosítom a DocumentBuilderFactory-t, majd normalizálom a dokumentumot. A dokumentum főbb elemeit NodeListekben tárolom el. A lekérdezés során egy NodeList-be lekérem az adott fő elem gyerekeit, majd egy for ciklussal végigiterálok a listán. Több táblás lekérdezés esetén dupla for ciklust alkalmazok.

```
package hu.domparse.IN3BLK;

import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;

public class DomQueryIN3BLK {

    public static void main(String[] args) {
        try {
            // XML fájl beolvasása
            File inputFile = new
File("XMLTaskIN3BLK\\DOMParseIN3BLK\\XMLIN3BLK.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            // 1. Lekérdezés: kiírja az 1-es ID-val rendelkező hotel adatait
            // Hotel elemek lekérdezése
            NodeList hotelList = doc.getElementsByTagName("Hotel");
```

```
            System.out.println("1. Lekérdezés:");

            // Hotel elemek bejárása
            for (int i = 0; i < hotelList.getLength(); i++) {
                Element hotelElement = (Element) hotelList.item(i);
                String h_kod = hotelElement.getAttribute("h_kod");

                // Ha a hotel kódja 1, akkor kiírja a hotel adatait
                if ("1".equals(h_kod)) {
                    System.out.println("Az 1-es ID-val rendelkező hotel
adatai:");
                    System.out.println("Név: " +
hotelElement.getElementsByTagName("Nev").item(0).getTextContent());
                    System.out.println(
                            "Telefon: " +
hotelElement.getElementsByTagName("Telefon").item(0).getTextContent());

                    // Cím elem lekérdezése
                    Element cimElement = (Element)
hotelElement.getElementsByTagName("Cim").item(0);
                    if (cimElement != null) {
                        System.out.println("Cím: " +
                                cimElement.getElementsByTagName("Irszam").item
(0).getTextContent() + ", " +
                                cimElement.getElementsByTagName("Varos").item(
0).getTextContent() + ", " +
                                cimElement.getElementsByTagName("Utca-
hazszam").item(0).getTextContent());
                    }

                    System.out.println(
                            "Értékelés: " +
hotelElement.getElementsByTagName("Ertekeles").item(0).getTextContent());
                    break; // Kilépü a ciklusból, mert megtaláltuk az első
illeszkedő hotelt
                }
            }

            // 2. Lekérdezés: kiírja azoknak az alkalmazottaknak a nevét,
akiknek a bére
            // több, mint 300000Ft
            // Alkalmazott elemek lekérdezése
            NodeList alkalmazottList =
doc.getElementsByTagName("Alkalmazott");
            System.out.println("\n2. Lekérdezés:");

            System.out.println("Alkalmazottak, akiknek a bére több, mint
300000Ft:");
            for (int i = 0; i < alkalmazottList.getLength(); i++) {
```

```
                Element alkalmazottElement = (Element)
alkalmazottList.item(i);
                int ber =
Integer.parseInt(alkalmazottElement.getElementsByTagName("Ber").item(0).getTex
tContent());

                // Ha a bér több, mint 300000, akkor kiírja az alkalmazott
nevét
                if (ber > 300000) {
                    String nev =
alkalmazottElement.getElementsByTagName("Nev").item(0).getTextContent();
                    System.out.println(nev);
                }
            }

            // 3. Lekérdezés: kiírja azoknak a Vevőknek az adatait akik a
Royal Hotelben
            // foglaltak szobát
            // Hotel elemek lekérdezése
            NodeList hotelListRoyal = doc.getElementsByTagName("Hotel");
            System.out.println("\n3. Lekérdezés:");

            System.out.println("Royal Hotelhez tartozó vevők adatai:");
            for (int i = 0; i < hotelListRoyal.getLength(); i++) {
                Element hotelElement = (Element) hotelListRoyal.item(i);
                String hotelNev =
hotelElement.getElementsByTagName("Nev").item(0).getTextContent();

                if ("Royal Hotel".equals(hotelNev)) {
                    String hotelKod = hotelElement.getAttribute("h_kod");

                    // Vevo elemek lekérdezése a megfelelő hotel kód alapján
                    NodeList vevoList = doc.getElementsByTagName("Vevo");
                    for (int j = 0; j < vevoList.getLength(); j++) {
                        Element vevoElement = (Element) vevoList.item(j);
                        String vevoSzKod = vevoElement.getAttribute("sz_kod");

                        if (vevoSzKod.equals(hotelKod)) {
                            System.out.println(
                                    "Név: " +
vevoElement.getElementsByTagName("Nev").item(0).getTextContent());
                            System.out.println(
                                    "Telefon: " +
vevoElement.getElementsByTagName("Telefon").item(0).getTextContent());
                            System.out.println(
                                    "Email: " +
vevoElement.getElementsByTagName("Email").item(0).getTextContent());
                        }
                    }
```

```
                }
            }

            // 4. lekérdezés: kiírja Kiss Anna számlázási adatait, azaz a
szálloda nevét, a
            // szoba emeletét, a foglalás kezdetét és végét, valamint a
fizetendő összeget
            // Vevő elemek lekérdezése "Kiss Anna" név alapján
            NodeList vevoList = doc.getElementsByTagName("Vevo");
            System.out.println("\n4. Lekérdezés:");

            for (int i = 0; i < vevoList.getLength(); i++) {
                Element vevoElement = (Element) vevoList.item(i);
                String vevoName =
vevoElement.getElementsByTagName("Nev").item(0).getTextContent();

                if ("Kiss Anna".equals(vevoName)) {
                    String customerSzKod = vevoElement.getAttribute("sz_kod");

                    // Hotel, Szoba, Foglalas és Szamla elemek lekérdezése a
vevő kódja alapján
                    NodeList hotelListSzamlazas =
doc.getElementsByTagName("Hotel");
                    for (int j = 0; j < hotelListSzamlazas.getLength(); j++) {
                        Element hotelElement = (Element)
hotelListSzamlazas.item(j);
                        String hotelKod = hotelElement.getAttribute("h_kod");

                        if (hotelKod.equals(customerSzKod)) {
                            System.out.println("Kiss Anna számlázási
adatai:");
                            System.out.println(
                                    "Hotel neve: " +
hotelElement.getElementsByTagName("Nev").item(0).getTextContent());

                            // Szoba és Foglalas elemek lekérdezése a szálloda
kódja alapján
                            NodeList szobaList =
doc.getElementsByTagName("Szoba");
                            for (int k = 0; k < szobaList.getLength(); k++) {
                                Element szobaElement = (Element)
szobaList.item(k);
                                String szobaSzKod =
szobaElement.getAttribute("sz_kod");

                                if (szobaSzKod.equals(hotelKod)) {
                                    System.out.println("Szoba emelete: "
                                            +
szobaElement.getElementsByTagName("Emelet").item(0).getTextContent());
```

```
                                    // Foglalas elemek lekérdezése a szoba
kódja alapján
                                    NodeList foglalasList =
doc.getElementsByTagName("Foglalas");
                                    for (int l = 0; l <
foglalasList.getLength(); l++) {
                                        Element foglalasElement = (Element)
foglalasList.item(l);
                                        String foglalasSzKod =
foglalasElement.getAttribute("sz_kod");

                                        if (foglalasSzKod.equals(szobaSzKod))
{
                                            System.out.println("Foglalás
kezdete: " + foglalasElement
                                                    .getElementsByTagName("Kez
det").item(0).getTextContent());
                                            System.out.println("Foglalás vége:
" + foglalasElement
                                                    .getElementsByTagName("Veg
e").item(0).getTextContent());
                                        }
                                    }

                                    // Szamla elemek lekérdezése a vevő kódja
alapján
                                    NodeList szamlaList =
doc.getElementsByTagName("Szamla");
                                    for (int m = 0; m <
szamlaList.getLength(); m++) {
                                        Element szamlaElement = (Element)
szamlaList.item(m);
                                        String szamlaVKod =
szamlaElement.getAttribute("v_kod");

                                        if (szamlaVKod.equals(customerSzKod))
{
                                            System.out.println("Fizetendő
összeg: " + szamlaElement
                                                    .getElementsByTagName("Oss
zeg").item(0).getTextContent());
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
```

```
            }

            // 5. lekérdezés: kiírja azoknak a vevőknek a nevét, akik
bankkártyával
            // fizettek, valamint a fizetés dátumát
            // Fizetes elemek lekérdezése Bankkártya fizetésmóddal
            NodeList fizetesList = doc.getElementsByTagName("Fizetes");
            System.out.println("\n5. Lekérdezés:");

            System.out.println("Vevők, akik Bankkártyával fizettek:");
            for (int i = 0; i < fizetesList.getLength(); i++) {
                Element fizetesElement = (Element) fizetesList.item(i);
                String fizetesmod =
fizetesElement.getElementsByTagName("Fizetesmod").item(0).getTextContent();

                if ("Bankkártya".equals(fizetesmod)) {
                    String vevoFizetesVKod =
fizetesElement.getAttribute("v_kod");

                    // Vevo elemek lekérdezése a vevő kódja alapján
                    NodeList vevokList = doc.getElementsByTagName("Vevo");
                    for (int j = 0; j < vevokList.getLength(); j++) {
                        Element vevoElement = (Element) vevokList.item(j);
                        String vevoVKod = vevoElement.getAttribute("v_kod");

                        if (vevoVKod.equals(vevoFizetesVKod)) {
                            System.out.println(
                                    "Vevő neve: " +
vevoElement.getElementsByTagName("Nev").item(0).getTextContent());

                            // Fizeteshez tartozó Dátum lekérdezése
                            System.out.println("Fizetés dátuma: "
                                    +
fizetesElement.getElementsByTagName("Datum").item(0).getTextContent() + "\n");
                        }
                    }
                }
            }

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

}
```

2d) adatírás

A három könyvtárak importálása után a fájl beolvasását végzem egy try-catch blokkban. Példányosítom a DocumentBuilderFactory-t, majd normalizálom a dokumentumot. A dokumentum főbb elemeit NodeListekben tárolom el. A dokumentum fastruktúrájának felépítését úgy valósítom meg, hogy létrehozom a root (gyökér) elemet, majd ehhez adom hozzá a később létrehozott főelemeket. Kiíratás a konzolra és a fájlba az adatolvasás (DOMRead) feladathoz hasonló módon történik.

```
package hu.domparse.IN3BLK;

import org.w3c.dom.*;
import java.io.*;

import javax.xml.parsers.*;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;

import java.util.Arrays;
import java.util.List;
import java.util.StringJoiner;

public class DomWriteIN3BLK {

    public static void main(String[] args) {
        try {
            // Dokumentum elkészítése
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.newDocument();

            // Gyökér elem létrehozása
            Element rootElement = doc.createElement("IN3BLK_Foglalas");
            rootElement.setAttribute("xmlns:xs",
"http://www.w3.org/2001/XMLSchema-instance");
            rootElement.setAttribute("xs:noNamespaceSchemaLocation",
"XMLSchemaIN3BLK.xsd");
            doc.appendChild(rootElement);

            // Szoba adatok hozzáadása
            addSzoba(doc, rootElement, "1", "2", "36000", "Szabad");
            addSzoba(doc, rootElement, "2", "5", "43000", "Szabad");
            addSzoba(doc, rootElement, "3", "1", "31000", "Foglalt");

            // Alkalmazott adatok hozzáadása
```

```
            addAlkalmazott(doc, rootElement, "1", "1", "Tóth András",
Arrays.asList("06-70-938-5617"), "320000",
                    "Portás");
            addAlkalmazott(doc, rootElement, "2", "2", "Lakatos Kevin",
                    Arrays.asList("06-30-294-7335", "06-20-993-3744"),
"390000", "Szakács");
            addAlkalmazott(doc, rootElement, "3", "3", "Szabó Ilona",
Arrays.asList("06-70-223-9485"), "280000",
                    "Takarító");

            // Hotel adatok hozzáadása
            addHotel(doc, rootElement, "1", "1", "Kényelem Hotel",
Arrays.asList("06-20-993-7766"), "1028", "Budapest",
                    "Szeles utca 25", "3");
            addHotel(doc, rootElement, "2", "2", "Royal Hotel",
Arrays.asList("06-70-234-3443"), "4033", "Debrecen",
                    "Erdei utca 6", "5");
            addHotel(doc, rootElement, "3", "3", "Udvari Hotel",
Arrays.asList("06-30-345-2345", "06-30-999-2384"),
                    "3521", "Miskolc", "Jakab utca 56", "4");

            // Számla adatok hozzáadása
            addSzamla(doc, rootElement, "1", "1", "Kiss Anna", "2020-02-17",
"86000");
            addSzamla(doc, rootElement, "2", "2", "Török András", "2021-09-
10", "113000");
            addSzamla(doc, rootElement, "3", "3", "Horváth Áron", "2022-05-
14", "93000");

            // Vevő adatok hozzáadása
            addVevo(doc, rootElement, "1", "1", "Kiss Anna",
Arrays.asList("06-30-222-2345", "06-70-399-5577"),
                    "kAnna@gmail.com");
            addVevo(doc, rootElement, "2", "2", "Török András",
Arrays.asList("06-20-948-3857", "06-30-947-5872"),
                    "tAndras@freemail.hu");
            addVevo(doc, rootElement, "3", "3", "Horváth Áron",
Arrays.asList("06-70-993-6665"), "hAron@gmail.com");

            // Foglalás adatok hozzáadása
            addFoglalas(doc, rootElement, "1", "1", "2020-02-10", "2020-02-
17");
            addFoglalas(doc, rootElement, "2", "2", "2021-09-03", "2021-09-
10");
            addFoglalas(doc, rootElement, "3", "3", "2022-05-10", "2022-05-
14");

            // Fizetés adatok hozzáadása
            addFizetes(doc, rootElement, "1", "1", "2020-02-17", "Készpénz");
```

```
            addFizetes(doc, rootElement, "2", "2", "2021-09-10",
"Bankkártya");
            addFizetes(doc, rootElement, "3", "3", "2022-05-14",
"Bankkártya");

            // Dokumentum mentése
            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            transformer.setOutputProperty("{https://xml.apache.org/xslt}indent
-amount", "2");

            printDocument(doc);

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

    private static void addSzoba(Document doc, Element rooElement, String
sz_kod, String emelet, String ar,
            String szabad) {
        // Szoba elem létrehozása
        Element szoba = doc.createElement("Szoba");
        // Szoba elem attribútumainak beállítása
        szoba.setAttribute("sz_kod", sz_kod);

        // Szoba elem gyerek elemeinek létrehozása
        Element emeletElement = createElement(doc, "Emelet", emelet);
        Element arElement = createElement(doc, "Ar", ar);
        Element szabadElement = createElement(doc, "Szabad", szabad);

        // Szoba elem gyerek elemeinek hozzáadása
        szoba.appendChild(emeletElement);
        szoba.appendChild(arElement);
        szoba.appendChild(szabadElement);

        // Szoba elem hozzáadása a gyökér elemhez
        rooElement.appendChild(szoba);
    }

    private static void addAlkalmazott(Document doc, Element rooElement,
String a_kod, String h_kod, String nev,
            List<String> telefonok, String ber, String beosztas) {
        Element alkalmazott = doc.createElement("Alkalmazott");
        alkalmazott.setAttribute("a_kod", a_kod);
        alkalmazott.setAttribute("h_kod", h_kod);
```

```
        Element nevElement = createElement(doc, "Nev", nev);
        for (String telefon : telefonok) {
            Element telefonElement = createElement(doc, "Telefon", telefon);
            alkalmazott.appendChild(telefonElement);
        }
        Element berElement = createElement(doc, "Ber", ber);
        Element beosztasElement = createElement(doc, "Beosztas", beosztas);

        alkalmazott.appendChild(nevElement);
        alkalmazott.appendChild(berElement);
        alkalmazott.appendChild(beosztasElement);

        rooElement.appendChild(alkalmazott);
    }

    private static void addHotel(Document doc, Element rooElement, String
h_kod, String sz_kod, String nev,
            List<String> telefonok, String irszam, String varos, String
utca_hazszam, String ertekeles) {
        Element hotel = doc.createElement("Hotel");
        hotel.setAttribute("h_kod", h_kod);
        hotel.setAttribute("sz_kod", sz_kod);

        Element nevElement = createElement(doc, "Nev", nev);
        for (String telefon : telefonok) {
            Element telefonElement = createElement(doc, "Telefon", telefon);
            hotel.appendChild(telefonElement);
        }

        Element cimElement = doc.createElement("Cim");
        Element irszamElement = createElement(doc, "Irszam", irszam);
        Element varosElement = createElement(doc, "Varos", varos);
        Element utca_hazszamElement = createElement(doc, "Utca-hazszam",
utca_hazszam);

        Element ertekelesElement = createElement(doc, "Ertekeles", ertekeles);

        cimElement.appendChild(irszamElement);
        cimElement.appendChild(varosElement);
        cimElement.appendChild(utca_hazszamElement);

        hotel.appendChild(nevElement);
        hotel.appendChild(cimElement);
        hotel.appendChild(ertekelesElement);

        rooElement.appendChild(hotel);
    }
```

```
    private static void addSzamla(Document doc, Element rooElement, String
szam_kod, String v_kod, String nev,
            String datum, String osszeg) {
        Element szamla = doc.createElement("Szamla");
        szamla.setAttribute("szam_kod", szam_kod);
        szamla.setAttribute("v_kod", v_kod);

        Element nevElement = createElement(doc, "Nev", nev);
        Element datumElement = createElement(doc, "Datum", datum);
        Element osszegElement = createElement(doc, "Osszeg", osszeg);

        szamla.appendChild(nevElement);
        szamla.appendChild(datumElement);
        szamla.appendChild(osszegElement);

        rooElement.appendChild(szamla);
    }

    private static void addVevo(Document doc, Element rooElement, String
v_kod, String sz_kod, String nev,
            List<String> telefonok, String email) {
        Element vevo = doc.createElement("Vevo");
        vevo.setAttribute("v_kod", v_kod);
        vevo.setAttribute("sz_kod", sz_kod);

        Element nevElement = createElement(doc, "Nev", nev);
        for (String telefon : telefonok) {
            Element telefonElement = createElement(doc, "Telefon", telefon);
            vevo.appendChild(telefonElement);
        }
        Element emailElement = createElement(doc, "Email", email);

        vevo.appendChild(nevElement);
        vevo.appendChild(emailElement);

        rooElement.appendChild(vevo);
    }

    private static void addFoglalas(Document doc, Element rooElement, String
sz_kod, String v_kod, String kezdet,
            String vege) {
        Element foglalas = doc.createElement("Foglalas");
        foglalas.setAttribute("sz_kod", sz_kod);
        foglalas.setAttribute("v_kod", v_kod);

        Element kezdetElement = createElement(doc, "Kezdet", kezdet);
        Element vegeElement = createElement(doc, "Vege", vege);

        foglalas.appendChild(kezdetElement);
```

```
        foglalas.appendChild(vegeElement);

        rooElement.appendChild(foglalas);
    }

    private static void addFizetes(Document doc, Element rooElement, String
v_kod, String szam_kod, String datum,
            String fizetesmod) {
        Element fizetes = doc.createElement("Fizetes");
        fizetes.setAttribute("v_kod", v_kod);
        fizetes.setAttribute("szam_kod", szam_kod);

        Element datumElement = createElement(doc, "Datum", datum);
        Element fizetesmodElement = createElement(doc, "Fizetesmod",
fizetesmod);

        fizetes.appendChild(datumElement);
        fizetes.appendChild(fizetesmodElement);

        rooElement.appendChild(fizetes);
    }

    // Dokumentum kiírása
    private static void printDocument(Document doc) {
        try {
            // Mentés fájlba
            File outputFile = new
File("XMLTaskIN3BLK\\DOMParseIN3BLK\\XMLIN3BLK1.xml");
            PrintWriter writer = new PrintWriter(new FileWriter(outputFile,
true));

            // XML gyökér elemének kiíratása a konzolra és fájlba
            Element rootElement = doc.getDocumentElement();
            String rootName = rootElement.getTagName();
            StringJoiner rootAttributes = new StringJoiner(" ");
            NamedNodeMap rootAttributeMap = rootElement.getAttributes();

            for (int i = 0; i < rootAttributeMap.getLength(); i++) {
                Node attribute = rootAttributeMap.item(i);
                rootAttributes.add(attribute.getNodeName() + "=\"" +
attribute.getNodeValue() + "\"");
            }

            System.out.print("<?xml version=\"1.0\" encoding=\"UTF-8\"?> \n");
            writer.print("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");

            System.out.print("<" + rootName + " " + rootAttributes.toString()
+ "> \n");
```

```
            writer.print("<" + rootName + " " + rootAttributes.toString() + ">
\n");

            NodeList szobaList = doc.getElementsByTagName("Szoba");
            NodeList alkalmazottList =
doc.getElementsByTagName("Alkalmazott");
            NodeList hotelList = doc.getElementsByTagName("Hotel");
            NodeList szamlaList = doc.getElementsByTagName("Szamla");
            NodeList vevoList = doc.getElementsByTagName("Vevo");
            NodeList foglalasList = doc.getElementsByTagName("Foglalas");
            NodeList fizetesList = doc.getElementsByTagName("Fizetes");

            // XML kiírása az eredeti formában
            System.out.println("");
            writer.println("");
            printNodeList(szobaList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(alkalmazottList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(hotelList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(szamlaList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(vevoList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(foglalasList, writer);
            System.out.println("");
            writer.println("");
            printNodeList(fizetesList, writer);

            // XML gyökér elemének lezárása
            System.out.println("</" + rootName + ">");
            writer.append("</" + rootName + ">");

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

    // NodeList kiírása
    private static void printNodeList(NodeList nodeList, PrintWriter writer) {
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
```

```
            printNode(node, 1, writer);
            System.out.println("");
            writer.println("");
        }
    }

    // Node kiírása
    private static void printNode(Node node, int indent, PrintWriter writer) {
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;
            String nodeName = element.getTagName();
            StringJoiner attributes = new StringJoiner(" ");
            NamedNodeMap attributeMap = element.getAttributes();

            for (int i = 0; i < attributeMap.getLength(); i++) {
                Node attribute = attributeMap.item(i);
                attributes.add(attribute.getNodeName() + "=\"" +
attribute.getNodeValue() + "\"");
            }

            System.out.print(getIndentString(indent));
            System.out.print("<" + nodeName + " " + attributes.toString() +
">");
            writer.print(getIndentString(indent));
            writer.print("<" + nodeName + " " + attributes.toString() + ">");

            NodeList children = element.getChildNodes();
            // Ellenőrzi, hogy az elemnek csak egy szöveges tartalma van-e
            if (children.getLength() == 1 && children.item(0).getNodeType() ==
Node.TEXT_NODE) {
                // Ha csak egy szöveges tartalom van, akkor kiíratja
                System.out.print(children.item(0).getNodeValue());
                writer.print(children.item(0).getNodeValue());
            } else {
                // Ha több gyerek eleme van, akkor új sor karaktereket és
behúzást ad hozzá
                System.out.println();
                writer.println();
                for (int i = 0; i < children.getLength(); i++) {
                    printNode(children.item(i), indent + 1, writer);
                }
                System.out.print(getIndentString(indent));
                writer.print(getIndentString(indent));
            }
            System.out.println("</" + nodeName + ">");
            writer.println("</" + nodeName + ">");
        }

    }
```

```
    private static Element createElement(Document doc, String name, String
value) {
        Element element = doc.createElement(name);
        element.appendChild(doc.createTextNode(value));
        return element;
    }

    // Behúzások létrehozása
    private static String getIndentString(int indent) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < indent; i++) {
            // Minden iteráció során két szóközt fűz hozzá a StringBuilderhez
            sb.append("  ");
        }
        return sb.toString();
    }

}
```