JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Számítógép üzlet

Készítette: Dobai Attila

Neptunkód: DIZ4VX

Dátum: 2023. 12. 02.

Tartalomjegyzék:

[**Feladat leírása:** 3](#_Toc152700573)

1.Feladat:

[**Az ER modell egyedei és tulajdonságai:** 3](#_Toc152700574)

[**Az XDM modellre konvertálás:** 5](#_Toc152700575)

[**Az XMLdokumentum készítése:** 6](#_Toc152700576)

[**XMLSchema készítése:** 11](#_Toc152700577)

2.Feladat:

[**Adatolvasás:** 18](#_Toc152700578)

[**Adatmódosítás:** 21](#_Toc152700579)

[**Adatlekérdezés:** 23](#_Toc152700580)

[**Adatírás:** 25](#_Toc152700581)

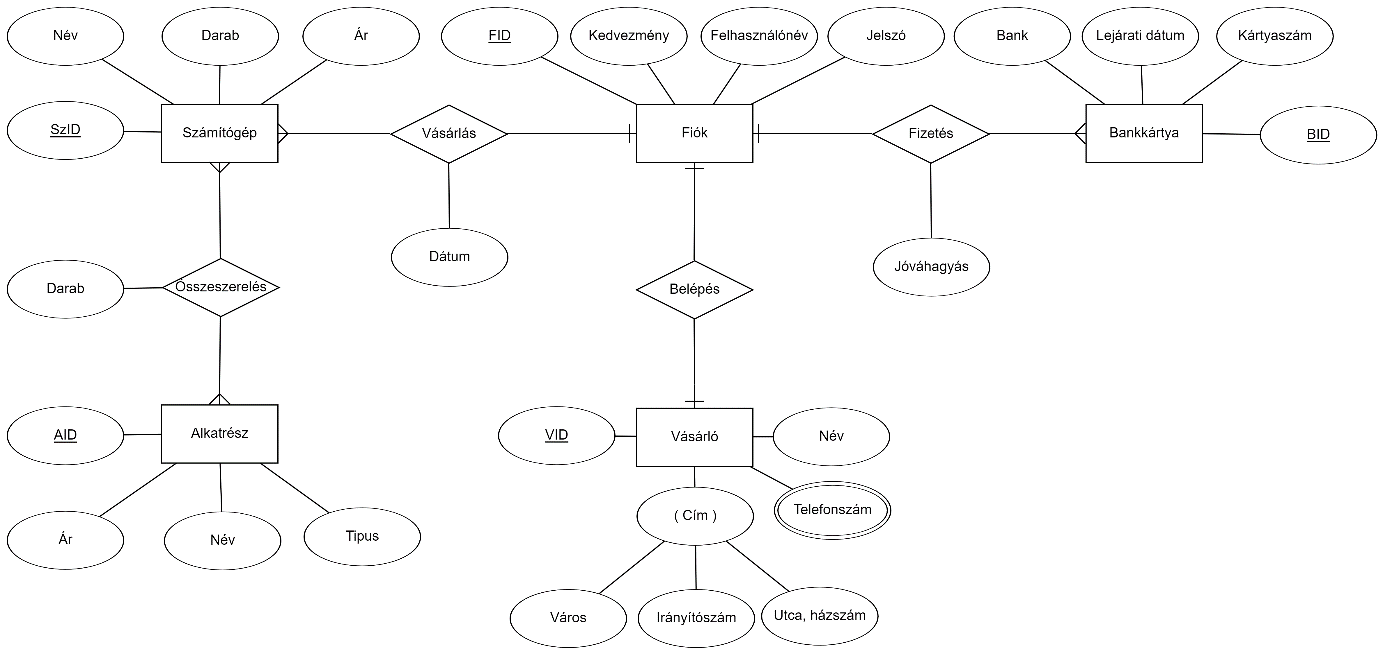
# **Feladat leírása:**

Az adatbázis témája egy számítógép üzlet, ahol bankkártával lehet egyszerre több gépet is megvásárolni. A vásárláshoz egy fiókot kell létrehozni, így akár kedvezményben is részesülhetünk. A számítógépek alkatrészekből épülnek fel, amiket eltároljuk az árukkal és a vásárlás dátumával együtt. Ezen felül a vásárlásnál eltároljuk a fizetés jóváhagyását.

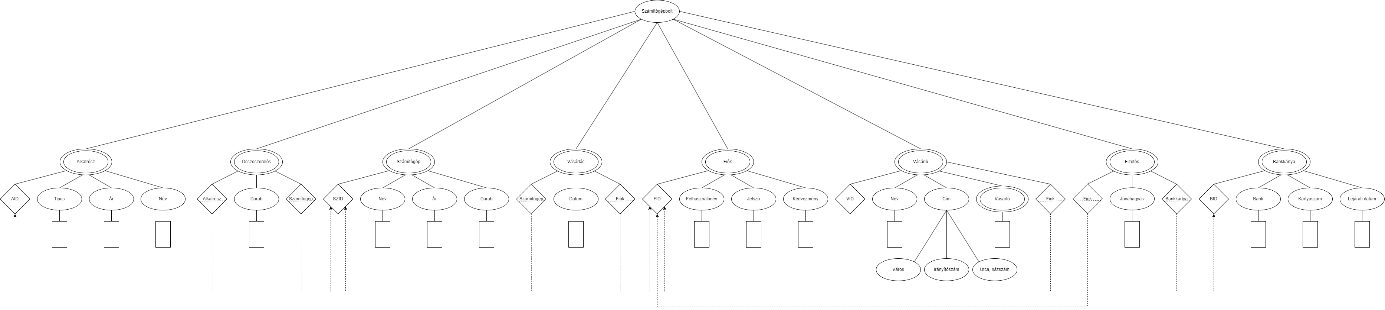
**Az adatbázis ER modell tervezése:**

# **Az ER modell egyedei és tulajdonságai:**

* **Alkatrészek**
  + **AID:** Az alkatrészek elsődleges kulcsa.
  + **Típus:** Az alkatrész típusa.
  + **Ár:** Az alkatrész ára.
  + **Név:** Az alkatrész neve.
* **Számítógép**
  + **SzID:** A számítógép elsődleges kulcsa.
  + **Név:** A számítógép neve.
  + **Ár:** A számítógép ára. Származtatott tulajdonság.
  + **Darab:** A számítógép mennyisége.
* **Vásárló**
  + **VID:** A vásárló elsődleges kulcsa.
  + **Név:** A vásárló neve.
  + **Telefonszám:** A vásárló telefonszáma. Többszörös tulajdonság.
  + **Cím:** A vásárló címe, ami egy összetett tulajdonság.
    - **Város**
    - **Irányítószám**
    - **Utca, házszám**
* **Fiók**
  + **FID:** A fiók elsődleges kulcsa.
  + **Felhasználónév:** A felhasználó neve.
  + **Jelszó:** A fiók jelszava.
  + **Kedvezmény:** A fiók kedvezménye, amire jogosult.
* **Bankkártya**
  + **BID:** A bankkártya elsődleges kulcsa.
  + **Bank:** A bank, ahova a bankkártya tartozik.
  + **Lejárati dátum:** A bankkártya lejárati dátuma.
  + **Kártyaszám:** A bankkártya száma.
* **Alkatrészek és Számítógép kapcsolata**
  + Ez egy több-több kapcsolat, mert egy számítógép több alkatrészből áll és vannak olyan alkatrészek, amelyekből csak több kell több géphez is kellenek. Ennek van 1 tulajdonsága, ami darab.
* **Számítógép és Vásárló kapcsolata**
  + Egy több kapcsolat, mert egy vásárló egyszerre vehet több számítógépet, de egy számítógépet, csak egy ember vehet meg. Ennek van 1 tulajdonsága, ami Dátum.
* **Vásárló és Bankkártya kapcsolata**
  + Egy több kapcsolat a kártya nem tartozhat több emberhez, de egy embernek lehet több bankkártyája is. Ennek van 1 tulajdonsága, ami Jóváhagyás.
* **Fiók és Vásárló kapcsolata**
  + Egy egy kapcsolat, mert egy vásárlónak csak 1 fiókja lehet.



# **Az XDM modellre konvertálás:**

Az adatbázis XDM modellje az ER modell alapján került megvalósításra. A gyökérelem neve a Számítógépbolt lett, amelynek 8db gyerekeleme van, amelyeknek mind van kulcs attribútuma. ****

# **Az XMLdokumentum készítése:**

Az XDM modell alapján az XML dokumentumot elkészítettem, minden elem 3 példánnyal van feltöltve.

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

<Számítógépbolt xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"

xs:noNamespaceSchemaLocation="XMLSchemaDIZ4VX.xsd">

<!-- Első Fiók -->

<Fiók FID="1">

<Felhasználónév>user1</Felhasználónév>

<Jelszó>password1</Jelszó>

<Kedvezmény>10</Kedvezmény>

</Fiók>

<!-- Második Fiók -->

<Fiók FID="2">

<Felhasználónév>user2</Felhasználónév>

<Jelszó>password2</Jelszó>

<Kedvezmény>15</Kedvezmény>

</Fiók>

<!-- Harmadik Fiók -->

<Fiók FID="3">

<Felhasználónév>user3</Felhasználónév>

<Jelszó>password3</Jelszó>

<Kedvezmény>20</Kedvezmény>

</Fiók>

<!-- Első Vásárló -->

<Vásárló VID="101" Fiók="1">

<Név>John Doe</Név>

<Cím>

<Város>Budapest</Város>

<Irányítószám>1111</Irányítószám>

<Utca\_házszám>Main Street 123</Utca\_házszám>

</Cím>

<Telefonszám>+36 30 123 4567</Telefonszám>

</Vásárló>

<!-- Második Vásárló -->

<Vásárló VID="102" Fiók="2">

<Név>Jane Doe</Név>

<Cím>

<Város>Pécs</Város>

<Irányítószám>2222</Irányítószám>

<Utca\_házszám>Second Street 456</Utca\_házszám>

</Cím>

<Telefonszám>+36 30 987 6543</Telefonszám>

</Vásárló>

<!-- Harmadik Vásárló -->

<Vásárló VID="103" Fiók="3">

<Név>Bob Smith</Név>

<Cím>

<Város>Szeged</Város>

<Irányítószám>3333</Irányítószám>

<Utca\_házszám>Third Street 789</Utca\_házszám>

</Cím>

<Telefonszám>+36 30 555 1234</Telefonszám>

</Vásárló>

<!-- Első Bankkártya -->

<Bankkártya BID="201">

<Bank>Bank of XYZ</Bank>

<Kártyaszám>1234-5678-9012-3456</Kártyaszám>

<Lejárati\_dátum>2025-12-31</Lejárati\_dátum>

</Bankkártya>

<!-- Második Bankkártya -->

<Bankkártya BID="202">

<Bank>Another Bank</Bank>

<Kártyaszám>9876-5432-1098-7654</Kártyaszám>

<Lejárati\_dátum>2024-08-15</Lejárati\_dátum>

</Bankkártya>

<!-- Harmadik Bankkártya -->

<Bankkártya BID="203">

<Bank>Bank XYZ Again</Bank>

<Kártyaszám>1111-2222-3333-4444</Kártyaszám>

<Lejárati\_dátum>2023-05-20</Lejárati\_dátum>

</Bankkártya>

<!-- Első Számítógép -->

<Számítógép SZID="301">

<Név>Gamer PC</Név>

<Ár>1500</Ár>

<Darab>2</Darab>

</Számítógép>

<!-- Második Számítógép -->

<Számítógép SZID="302">

<Név>Office PC</Név>

<Ár>800</Ár>

<Darab>1</Darab>

</Számítógép>

<!-- Harmadik Számítógép -->

<Számítógép SZID="303">

<Név>Developer PC</Név>

<Ár>1200</Ár>

<Darab>3</Darab>

</Számítógép>

<!-- Első Alkatrész -->

<Alkatrész AID="401">

<Név>RAM modul</Név>

<Ár>100</Ár>

<Tipus>DDR4</Tipus>

</Alkatrész>

<!-- Második Alkatrész -->

<Alkatrész AID="402">

<Név>SSD meghajtó</Név>

<Ár>120</Ár>

<Tipus>SATA</Tipus>

</Alkatrész>

<!-- Harmadik Alkatrész -->

<Alkatrész AID="403">

<Név>Videokártya</Név>

<Ár>300</Ár>

<Tipus>GPU</Tipus>

</Alkatrész>

<!-- Első Fizetés -->

<Fizetés fiók="1" bankkártya="201">

<Jóváhagyás>Igen</Jóváhagyás>

</Fizetés>

<!-- Második Fizetés -->

<Fizetés fiók="2" bankkártya="202">

<Jóváhagyás>Nem</Jóváhagyás>

</Fizetés>

<!-- Harmadik Fizetés -->

<Fizetés fiók="3" bankkártya="203">

<Jóváhagyás>Igen</Jóváhagyás>

</Fizetés>

<!-- Első Vásárlás -->

<Vásárlás fiók="1" számítógép="301">

<Dátum>2023-11-18</Dátum>

</Vásárlás>

<!-- Második Vásárlás -->

<Vásárlás fiók="2" számítógép="302">

<Dátum>2023-11-19</Dátum>

</Vásárlás>

<!-- Harmadik Vásárlás -->

<Vásárlás fiók="3" számítógép="303">

<Dátum>2023-11-20</Dátum>

</Vásárlás>

<!-- Első Összeszerelés -->

<Összeszerelés számítógép="301" alkatrész="401">

<Darab>2</Darab>

</Összeszerelés>

<!-- Második Összeszerelés -->

<Összeszerelés számítógép="302" alkatrész="402">

<Darab>1</Darab>

</Összeszerelés>

<!-- Harmadik Összeszerelés -->

<Összeszerelés számítógép="303" alkatrész="403">

<Darab>3</Darab>

</Összeszerelés>

</Számítógépbolt>

# **XMLSchema készítése:**

Az XML dokumentum validálására XMLSchema sémaleíró dokumentumot hoztam létre. Alkalmaztam benne különféle megszorításokat, komplex tipusokat hoztam létre, elsődleges, idegenkulcsokkal együtt. A sémát a VS code sikeresen validálta.

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

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

<!-- Az entitások leírása -->

<!-- Felhasználó adatai -->

<xs:element name="Felhasználónév" type="xs:string" />

<xs:element name="Jelszó" type="xs:string" />

<xs:element name="Kedvezmény" type="xs:int" />

<!-- Vásárló adatai -->

<xs:element name="Név" type="xs:string" />

<xs:element name="Telefonszám" type="xs:string" />

<xs:element name="Bank" type="xs:string" />

<xs:element name="Kártyaszám" type="kártyaszámTípus" />

<xs:element name="Lejárati\_dátum" type="dátumTípus" />

<!-- Termék adatai -->

<xs:element name="Ár" type="xs:int" />

<xs:element name="Darab" type="xs:int" />

<xs:element name="Tipus" type="xs:string" />

<!-- Egyéb adatok -->

<xs:element name="Jóváhagyás" type="xs:string" />

<xs:element name="Dátum" type="dátumTípus" />

<!-- A speciális típusok definiálása -->

<xs:simpleType name="irányítószámTípus">

<xs:restriction base="xs:int">

<xs:pattern value="(\d{4})" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="kártyaszámTípus">

<xs:restriction base="xs:string">

<xs:pattern value="(\d{4})-(\d{4})-(\d{4})-(\d{4})" />

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="dátumTípus">

<xs:restriction base="xs:string">

<xs:pattern value="(\d{4})-(\d{2})-(\d{2})" />

</xs:restriction>

</xs:simpleType>

<!-- A komplex típusok definiálása -->

<xs:complexType name="fiókTípus">

<xs:sequence>

<xs:element ref="Felhasználónév" />

<xs:element ref="Jelszó" />

<xs:element ref="Kedvezmény" />

</xs:sequence>

<xs:attribute name="FID" type="xs:integer" use="required" />

</xs:complexType>

<xs:complexType name="vásárlóTípus">

<xs:sequence>

<xs:element ref="Név" />

<xs:element name="Cím">

<xs:complexType>

<xs:sequence>

<xs:element name="Város" type="xs:string" />

<xs:element name="Irányítószám" type="irányítószámTípus" />

<xs:element name="Utca\_házszám" type="xs:string" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element ref="Telefonszám" />

</xs:sequence>

<xs:attribute name="VID" type="xs:integer" use="required" />

<xs:attribute name="Fiók" type="xs:integer" use="required" />

</xs:complexType>

<xs:complexType name="bankkártyaTípus">

<xs:sequence>

<xs:element ref="Bank" />

<xs:element ref="Kártyaszám" />

<xs:element ref="Lejárati\_dátum" />

</xs:sequence>

<xs:attribute name="BID" type="xs:integer" use="required" />

</xs:complexType>

<xs:complexType name="számítógépTípus">

<xs:sequence>

<xs:element ref="Név" />

<xs:element ref="Ár" />

<xs:element ref="Darab" />

</xs:sequence>

<xs:attribute name="SZID" type="xs:integer" use="required" />

</xs:complexType>

<xs:complexType name="alkatrészTípus">

<xs:sequence>

<xs:element ref="Név" />

<xs:element ref="Ár" />

<xs:element ref="Tipus" />

</xs:sequence>

<xs:attribute name="AID" type="xs:integer" use="required" />

</xs:complexType>

<xs:complexType name="fizetésTípus">

<xs:sequence>

<xs:element ref="Jóváhagyás" />

</xs:sequence>

<xs:attribute name="fiók" type="xs:integer" use="required" />

<xs:attribute name="bankkártya" type="xs:integer" use="required" />

</xs:complexType>

<xs:complexType name="vásárlásTípus">

<xs:sequence>

<xs:element ref="Dátum" />

</xs:sequence>

<xs:attribute name="fiók" type="xs:integer" use="required" />

<xs:attribute name="számítógép" type="xs:integer" use="required" />

</xs:complexType>

<xs:complexType name="összeszerelésTípus">

<xs:sequence>

<xs:element ref="Darab" />

</xs:sequence>

<xs:attribute name="számítógép" type="xs:integer" use="required" />

<xs:attribute name="alkatrész" type="xs:integer" use="required" />

</xs:complexType>

<!-- A Számítógépbolt fő entitása -->

<xs:element name="Számítógépbolt">

<xs:complexType>

<xs:sequence>

<xs:element name="Fiók" type="fiókTípus" minOccurs="0" maxOccurs="unbounded" />

<xs:element name="Vásárló" type="vásárlóTípus" minOccurs="0" maxOccurs="unbounded" />

<xs:element name="Bankkártya" type="bankkártyaTípus" minOccurs="0"

maxOccurs="unbounded" />

<xs:element name="Számítógép" type="számítógépTípus" minOccurs="0"

maxOccurs="unbounded" />

<xs:element name="Alkatrész" type="alkatrészTípus" minOccurs="0"

maxOccurs="unbounded" />

<xs:element name="Fizetés" type="fizetésTípus" minOccurs="0" maxOccurs="unbounded" />

<xs:element name="Vásárlás" type="vásárlásTípus" minOccurs="0" maxOccurs="unbounded" />

<xs:element name="Összeszerelés" type="összeszerelésTípus" minOccurs="0"

maxOccurs="unbounded" />

</xs:sequence>

</xs:complexType>

<!-- Kulcsok és kulcsreferenciák definiálása -->

<xs:key name="fiók\_kulcs">

<xs:selector xpath="Fiók" />

<xs:field xpath="@FID" />

</xs:key>

<xs:key name="vásárló\_kulcs">

<xs:selector xpath="Vásárló" />

<xs:field xpath="@VID" />

</xs:key>

<xs:key name="bankkártya\_kulcs">

<xs:selector xpath="Bankkártya" />

<xs:field xpath="@BID" />

</xs:key>

<xs:key name="számítógép\_kulcs">

<xs:selector xpath="Számítógép" />

<xs:field xpath="@SZID" />

</xs:key>

<xs:key name="alkatrész\_kulcs">

<xs:selector xpath="Alkatrész" />

<xs:field xpath="@AID" />

</xs:key>

<xs:keyref name="fiók\_vásárló\_kulcs" refer="fiók\_kulcs">

<xs:selector xpath="Vásárló" />

<xs:field xpath="@fiók" />

</xs:keyref>

<xs:keyref name="fiók\_fizetés\_kulcs" refer="fiók\_kulcs">

<xs:selector xpath="Fizetés" />

<xs:field xpath="@fiók" />

</xs:keyref>

<xs:keyref name="bankkártya\_fizetés\_kulcs" refer="bankkártya\_kulcs">

<xs:selector xpath="Bankkártya" />

<xs:field xpath="@bankkártya" />

</xs:keyref>

<xs:keyref name="fiók\_vásárlás\_kulcs" refer="fiók\_kulcs">

<xs:selector xpath="Fiók" />

<xs:field xpath="@fiók" />

</xs:keyref>

<xs:keyref name="számítógép\_vásárlás\_kulcs" refer="számítógép\_kulcs">

<xs:selector xpath="Számítógép" />

<xs:field xpath="@számítógép" />

</xs:keyref>

<xs:keyref name="összeszerelés\_számítógép\_kulcs" refer="számítógép\_kulcs">

<xs:selector xpath="Számítógép" />

<xs:field xpath="@számítógép" />

</xs:keyref>

<xs:keyref name="alkatrész\_összeszerelés\_kulcs" refer="alkatrész\_kulcs">

<xs:selector xpath="Alkatrész" />

<xs:field xpath="@alkatrész" />

</xs:keyref>

</xs:element>

</xs:schema>

**2.Feladat:**

# **Adatolvasás:**

package hu.domparse.DIZ4VX;

import org.w3c.dom.Document;

import org.w3c.dom.NamedNodeMap;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import java.io.File;

import java.io.FileOutputStream;

import java.io.OutputStream;

import java.util.Properties;

public class DomReadDIZ4VX {

public static void main(String[] args) {

try {

// XML-dokumentum beolvasása

File xmlFile = new File("XMLDIZ4VX.xml");

Document doc = parseXML(xmlFile);

// A dokumentum fastruktúrájának kilistázása a konzolra

System.out.println("Fa struktúra:");

listNodes(doc.getDocumentElement(), "");

// Az új XML-fájl elkészítése

File outputFile = new File("XMLDIZ4VX1.xml");

writeXML(doc, outputFile);

System.out.println("A XMLDIZ4VX.xml fájl elkészült.");

} catch (Exception e) {

e.printStackTrace();

}

}

// Rekurzív módon kilistázza a dokumentum fastruktúráját

public static void listNodes(Node node, String indent) {

// Nyitó címke kiírása attribútumokkal

System.out.print(indent + "<" + node.getNodeName());

NamedNodeMap attributes = node.getAttributes();

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

Node attribute = attributes.item(i);

System.out.print(" " + attribute.getNodeName() + "=\"" + attribute.getNodeValue() + "\"");

}

// Szöveges tartalom kiírása, ha van

if (node.hasChildNodes()) {

NodeList childNodes = node.getChildNodes();

// Ellenőrizze, hogy a gyermek elemek között van-e ELEMENT\_NODE

boolean hasElementChild = false;

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

if (childNodes.item(i).getNodeType() == Node.ELEMENT\_NODE) {

hasElementChild = true;

break;

}

}

if (hasElementChild) {

System.out.println(">");

// Rekurzív hívás a gyermek elemekre

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

Node childNode = childNodes.item(i);

if (childNode.getNodeType() == Node.ELEMENT\_NODE) {

listNodes(childNode, indent + " ");

}

}

System.out.println(indent + "</" + node.getNodeName() + ">");

} else {

// Ha nincs más gyermek elem, akkor kiírja a szöveget és a záró címkét

String text = node.getTextContent().trim();

if (!text.isEmpty()) {

System.out.println(">" + text + "</" + node.getNodeName() + ">");

} else {

System.out.println("/>");

}

}

} else {

// Ha nincs gyermek eleme, záró címke zárással fejezzük be

System.out.println("/>");

}

}

// XML dokumentum beolvasása

public static Document parseXML(File file) throws Exception {

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

return dBuilder.parse(file);

}

// XML dokumentum kiírása fájlba strukturált formában

public static void writeXML(Document doc, File file) throws Exception {

TransformerFactory transformerFactory = TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

Properties outputProperties = new Properties();

outputProperties.setProperty(OutputKeys.INDENT, "yes");

transformer.setOutputProperties(outputProperties);

DOMSource source = new DOMSource(doc);

OutputStream os = new FileOutputStream(file);

StreamResult result = new StreamResult(os);

transformer.transform(source, result);

}

}

# **Adatmódosítás:**

package hu.domparse.DIZ4VX;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NodeList;

import java.io.File;

public class DomModifyDIZ4VX {

public static void main(String[] args) {

try {

// XML-dokumentum beolvasása

File xmlFile = new File("XMLDIZ4VX.xml");

Document doc = DomReadDIZ4VX.parseXML(xmlFile);

// Példa: Adatmódosítás (jelszó megváltoztatása)

modifyData(doc, "Fiók", "Felhasználónév", "user1", "Jelszó", "newPassword1");

modifyData(doc, "Fiók", "Felhasználónév", "user2", "Jelszó", "newPassword2");

modifyData(doc, "Fiók", "Felhasználónév", "user3", "Jelszó", "newPassword3");

modifyData(doc, "Számítógép", "Név", "Developer PC", "Darab", "5");

modifyData(doc, "Számítógép", "Név", "Office PC", "Darab", "50");

// Módosított dokumentum kiírása a konzolra

System.out.println("Módosított dokumentum:");

DomReadDIZ4VX.listNodes(doc.getDocumentElement(), "");

// Az új XML-fájl elkészítése

File outputFile = new File("ModifiedXMLDIZ4VX.xml");

DomReadDIZ4VX.writeXML(doc, outputFile);

System.out.println("A ModifiedXMLDIZ4VX.xml fájl elkészült.");

} catch (Exception e) {

e.printStackTrace();

}

}

// Általános adatmódosító metódus

private static void modifyData(Document doc, String tableName, String identifierTag, String identifierValue,

String fieldToModify, String newValue) {

NodeList nodeList = doc.getElementsByTagName(tableName);

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

Element element = (Element) nodeList.item(i);

String currentIdentifier = element.getElementsByTagName(identifierTag).item(0).getTextContent();

if (currentIdentifier.equals(identifierValue)) {

// Adat módosítása

element.getElementsByTagName(fieldToModify).item(0).setTextContent(newValue);

System.out.println("Adat módosítva: " + currentIdentifier + ", " + fieldToModify);

return; // Kilépés, ha a módosítás megtörtént

}

}

// Ha az azonosítót nem találjuk

System.out.println("Azonosító nem található: " + identifierValue);

}

}

# **Adatlekérdezés:**

package hu.domparse.DIZ4VX;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NodeList;

import java.io.File;

public class DomQueryDIZ4VX {

public static void main(String[] args) {

try {

// XML-dokumentum beolvasása

File xmlFile = new File("XMLDIZ4VX.xml");

Document doc = DomReadDIZ4VX.parseXML(xmlFile);

// Példa lekérdezések

System.out.println("Lekérdezések:");

// Lekérdezés 1:

queryData(doc, "Vásárló", "VID", "101", "Név");

// Lekérdezés 2:

queryData(doc, "Vásárló", "VID", "102", "Cím");

// Lekérdezés 3:

queryData(doc, "Számítógép", "SZID", "303", "Ár");

// Lekérdezés 4:

queryData(doc, "Számítógép", "SZID", "302", "Darab");

// Lekérdezés 5:

queryData(doc, "Fiók", "FID", "3", "Felhasználónév");

} catch (Exception e) {

e.printStackTrace();

}

}

// Általános adatlekérdező metódus

private static void queryData(Document doc, String tableName, String identifierAttribute, String identifierValue,

String fieldToQuery) {

NodeList nodeList = doc.getElementsByTagName(tableName);

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

Element element = (Element) nodeList.item(i);

// Azonosító attribútum lekérése

String currentIdentifier = element.getAttribute(identifierAttribute);

if (currentIdentifier.equals(identifierValue)) {

// Ellenőrzés, hogy a fieldToQuery létezik

NodeList fieldNodes = element.getElementsByTagName(fieldToQuery);

if (fieldNodes.getLength() > 0) {

// Adat lekérdezése

String queryResult = fieldNodes.item(0).getTextContent();

System.out.println("Lekérdezés eredménye: " + currentIdentifier + ", " + fieldToQuery + ": " + queryResult);

} else {

System.out.println("A mező nem található: " + fieldToQuery);

}

return; // Kilépés, ha a lekérdezés megtörtént

}

}

// Ha az azonosítót nem találjuk

System.out.println("Azonosító nem található: " + identifierValue);

}

}

# **Adatírás:**

package hu.domparse.DIZ4VX;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import java.io.File;

public class DOMWriteDIZ4VX {

public static void main(String[] args) {

try {

// Új dokumentum létrehozása

Document doc = createSampleDocument();

// A dokumentum fastruktúrájának kilistázása a konzolra

System.out.println("Fa struktúra:");

DomReadDIZ4VX.listNodes(doc.getDocumentElement(), "");

// Az új XML-fájl elkészítése

File outputFile = new File("XMLDIZ4VX2.xml");

DomReadDIZ4VX.writeXML(doc, outputFile);

System.out.println("A XMLDIZ4VX1.xml fájl elkészült.");

} catch (Exception e) {

e.printStackTrace();

}

}

// Új dokumentum létrehozása előre meghatározott adatokkal

private static Document createSampleDocument() throws Exception {

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.newDocument();

// Gyökér elem létrehozása

Element rootElement = doc.createElement("Fiók");

rootElement.setAttribute("FID", "1");

doc.appendChild(rootElement);

// Felhasználónév elem hozzáadása

Element felhasznalonevElement = doc.createElement("Felhasználónév");

felhasznalonevElement.setTextContent("user1");

rootElement.appendChild(felhasznalonevElement);

// Jelszó elem hozzáadása

Element jelszoElement = doc.createElement("Jelszó");

jelszoElement.setTextContent("password1");

rootElement.appendChild(jelszoElement);

// Kedvezmény elem hozzáadása

Element kedvezmenyElement = doc.createElement("Kedvezmény");

kedvezmenyElement.setTextContent("10");

rootElement.appendChild(kedvezmenyElement);

return doc;

}

}