JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Számítógép üzlet

Készítette: Dobai Attila

Neptunkód: DIZ4VX

Dátum: 2023. 12. 02.

Tartalomjegyzék

Bevezetés

A 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.

1.Feladat

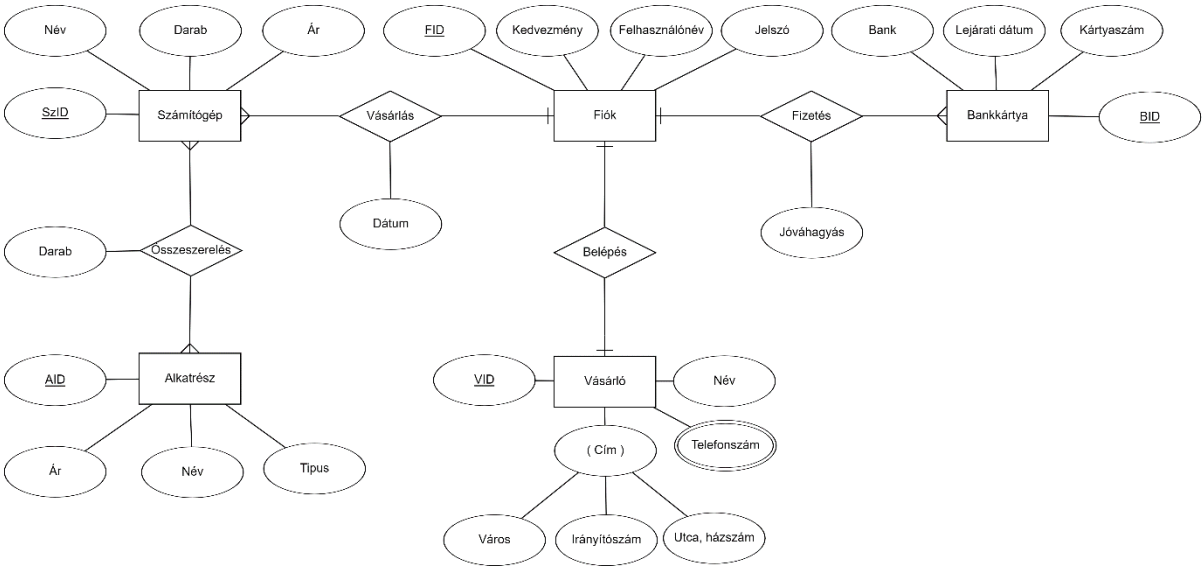
Az adatbázis ER modell tervezése

A feladat leírása alapján létrehozok egy ER modellt, ami ezt a számítógép boltot ki tudja szolgálni.

Az ER modell egyedei és tulajdonságai:

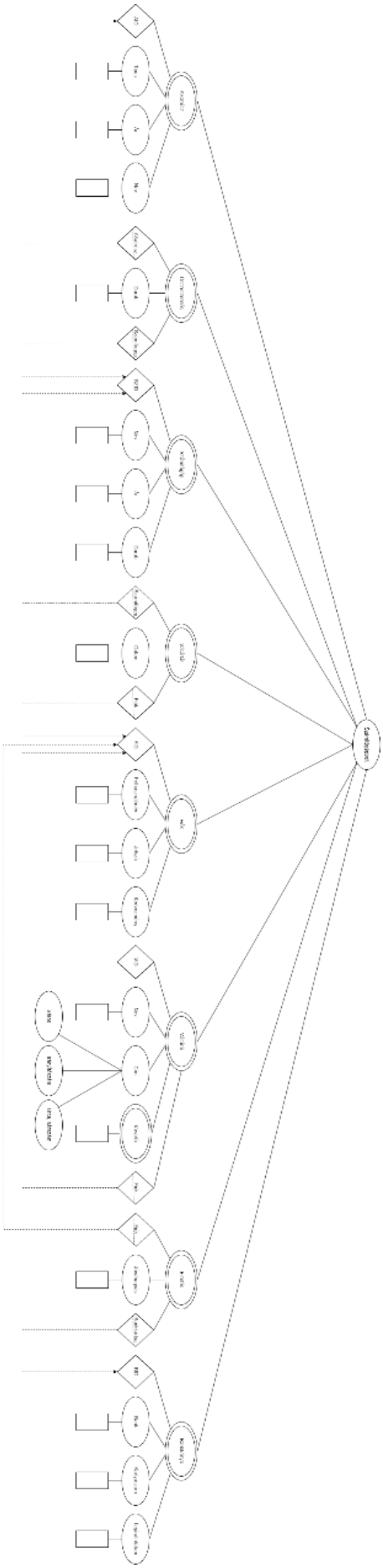
- **Alkatrészek**
  - **AID:** Az alkatrészek elsődleges kulcsa.
  - **Típus:** Az alkatrész típusa.
  - **Ár:** Az alkatrész ára.
  - **Név:** Az alkatrész neve.
- **Számítógép**
  - **SzID:** A számítógép elsődleges kulcsa.
  - **Név:** A számítógép neve.
  - **Ár:** A számítógép ára. Származtatott 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>
    <Telefonszám>+36 30 123 4568</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>
    <Telefonszám>+36 30 555 1235</Telefonszám>
    <Telefonszám>+36 30 555 1236</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" minOccurs="1" maxOccurs="3"/>
  </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>

<!-- Idegen kulcsok -->
<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>

<!-- Az 1:1 kapcsolat megvalósítása -->
```

```
      <xs:unique name="Vásárló_fiók_1_1">
        <xs:selector xpath="Vásárló" />
        <xs:field xpath="@fiók" />
      </xs:unique>

  </xs:element>

</xs:schema>
```

2.Feladat

Adatolvasás

Az xml beolvasása majd kiíratása strukturáltan konzolra, és a tartalma új fileként való mentése.

```
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

Az xml beolvasása, 5db elemnek az adatának a módosítása, utána kiíratása strukturáltan konzolra, a módosított xml új fileként való mentése.

```
package hu.domparse.DIZ4VX;


import org.w3c.dom.*;


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 DomModifyDIZ4VX {
   public static void main(String[] args) {
      try {
         // XML-dokumentum beolvasása
         File xmlFile = new File("XMLDIZ4VX.xml");
         Document doc = parseXML(xmlFile);


         // Példa: Adatmódosítás
         modifyData(doc, "Fiók", "Felhasználónév", "user1", "Jelszó", "newPassword1");
```

```
        modifyData(doc, "Bankkártya", "Kártyaszám", "1234-5678-9012-3456", "Lejárati_dátum",
"2028-10-20");
        modifyData(doc, "Alkatrész", "Név", "RAM modul", "Ár", "11500");
        modifyData(doc, "Számítógép", "Név", "Developer PC", "Darab", "5");
        modifyData(doc, "Számítógép", "Név", "Office PC", "Ár", "50000");

        // Módosított dokumentum kiírása a konzolra
        System.out.println("Módosított dokumentum:");
        listNodes(doc.getDocumentElement(), "");

        // Az új XML-fájl elkészítése
        File outputFile = new File("ModifiedXMLDIZ4VX.xml");
        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);
}
// 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);
    }
}
```

Adatlekérdezés

Itt az xml-ből 5db lekérdezést valósítok meg, amelyek erdményét strukturáltan kiírom a konzolra.

```
package hu.domparse.DIZ4VX;

import org.w3c.dom.*;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
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 = parseXML(xmlFile);

            // Példa lekérdezések
            System.out.println("Lekérdezések:");

            // Lekérdezés 1:
            queryData(doc, "Vásárló", "VID", "101");
            // Lekérdezés 2:
            queryData(doc, "Vásárló", "VID", "102");
            // Lekérdezés 3:
            queryData(doc, "Számítógép", "SZID", "303");
            // Lekérdezés 4:
            queryData(doc, "Számítógép", "SZID", "302");
            // Lekérdezés 5:
            queryData(doc, "Fiók", "FID", "3");

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

    // Általános adatlekérdező metódus
    private static void queryData(Document doc, String tableName, String identifierAttribute, String
identifierValue) {
        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)) {
                // Az összes adat kiírása az adott elemhez
                System.out.println("Lekérdezés eredménye:");
                listNodes(element, "  ");
                return; // Kilépés, ha a lekérdezés megtörtént
            }
        }

        // Ha az azonosítót nem találjuk
        System.out.println("Azonosító nem található: " + identifierValue);
    }

    // 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);
    }
}
```

Adatírás

A számítógép bolt xml tartalmának kézzel való felvitele, konzolra való kiírása, xml-ként való mentése.

```
package hu.domparse.DIZ4VX;

import org.w3c.dom.*;

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

            // A fájl neve és elérési útja
            File outputFile = new File("XMLDIZ4VX2.xml");
            // XML fájl írása
            writeXML(doc, outputFile);

            System.out.println("Az XMLDIZ4VX2.xml fájl elkészült.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static Document createSampleDocument() throws Exception {
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        Document doc = dBuilder.newDocument();

        // Gyökér elem létrehozása
        Element szamitogepboltElement = doc.createElement("Számítógépbolt");
        szamitogepboltElement.setAttribute("xmlns:xs", "http://www.w3.org/2001/XMLSchema-
instance");
        szamitogepboltElement.setAttribute("xs:noNamespaceSchemaLocation",
"XMLSchemaDIZ4VX.xsd");
        doc.appendChild(szamitogepboltElement);
```

```
        // Fiók elemek létrehozása
        szamitogepboltElement.appendChild(createFiokElement(doc, "1", "user1", "password1", "10"));
        szamitogepboltElement.appendChild(createFiokElement(doc, "2", "user2", "password2", "15"));
        szamitogepboltElement.appendChild(createFiokElement(doc, "3", "user3", "password3", "20"));

        // Vásárló elemek létrehozása
        szamitogepboltElement.appendChild(createVasarloElement(doc, "101", "1", "John Doe",
"Budapest", "1111", "Main Street 123", "+36 30 123 4567", "+36 30 123 4568"));
        szamitogepboltElement.appendChild(createVasarloElement(doc, "102", "2", "Jane Doe", "Pécs",
"2222", "Second Street 456", "+36 30 987 6543"));
        szamitogepboltElement.appendChild(createVasarloElement(doc, "103", "3", "Bob Smith",
"Szeged", "3333", "Third Street 789", "+36 30 555 1234", "+36 30 555 1235", "+36 30 555 1236"));

        // Bankkártya elemek létrehozása
        szamitogepboltElement.appendChild(createBankkartyaElement(doc, "201", "Bank of XYZ",
"1234-5678-9012-3456", "2025-12-31"));
        szamitogepboltElement.appendChild(createBankkartyaElement(doc, "202", "Another Bank",
"9876-5432-1098-7654", "2024-08-15"));
        szamitogepboltElement.appendChild(createBankkartyaElement(doc, "203", "Bank XYZ Again",
"1111-2222-3333-4444", "2023-05-20"));

        // Számítógép elemek létrehozása
        szamitogepboltElement.appendChild(createSzamitogepElement(doc, "301", "Gamer PC", "1500",
"2"));
        szamitogepboltElement.appendChild(createSzamitogepElement(doc, "302", "Office PC", "800",
"1"));
        szamitogepboltElement.appendChild(createSzamitogepElement(doc, "303", "Developer PC",
"1200", "3"));

        // Alkatrész elemek létrehozása
        szamitogepboltElement.appendChild(createAlkatreszElement(doc, "401", "RAM modul", "100",
"DDR4"));
        szamitogepboltElement.appendChild(createAlkatreszElement(doc, "402", "SSD meghajtó",
"120", "SATA"));
```

```
        szamitogepboltElement.appendChild(createAlkatreszElement(doc, "403", "Videokártya", "300",
"GPU"));

        // Fizetés elemek létrehozása
        szamitogepboltElement.appendChild(createFizetesElement(doc, "1", "201", "Igen"));
        szamitogepboltElement.appendChild(createFizetesElement(doc, "2", "202", "Nem"));
        szamitogepboltElement.appendChild(createFizetesElement(doc, "3", "203", "Igen"));

        // Vásárlás elemek létrehozása
        szamitogepboltElement.appendChild(createVasarlasElement(doc, "1", "301", "2023-11-18"));
        szamitogepboltElement.appendChild(createVasarlasElement(doc, "2", "302", "2023-11-19"));
        szamitogepboltElement.appendChild(createVasarlasElement(doc, "3", "303", "2023-11-20"));

        // Összeszerelés elemek létrehozása
        szamitogepboltElement.appendChild(createOsszeszerelésElement(doc, "301", "401", "2"));
        szamitogepboltElement.appendChild(createOsszeszerelésElement(doc, "302", "402", "1"));
        szamitogepboltElement.appendChild(createOsszeszerelésElement(doc, "303", "403", "3"));

        return doc;
    }

    private static Element createFiokElement(Document doc, String fid, String username, String
password, String kedvezmeny) {
        Element fiokElement = doc.createElement("Fiók");
        fiokElement.setAttribute("FID", fid);

        fiokElement.appendChild(createTextElement(doc, "Felhasználónév", username));
        fiokElement.appendChild(createTextElement(doc, "Jelszó", password));
        fiokElement.appendChild(createTextElement(doc, "Kedvezmény", kedvezmeny));

        return fiokElement;
    }
```

```
    private static Element createVasarloElement(Document doc, String vid, String fiok, String nev,
String varos, String iranyitoszam, String utcaHazszam, String... telefon) {
        Element vasarloElement = doc.createElement("Vásárló");
        vasarloElement.setAttribute("VID", vid);
        vasarloElement.setAttribute("Fiók", fiok);

        vasarloElement.appendChild(createTextElement(doc, "Név", nev));

        Element cimElement = doc.createElement("Cím");
        cimElement.appendChild(createTextElement(doc, "Város", varos));
        cimElement.appendChild(createTextElement(doc, "Irányítószám", iranyitoszam));
        cimElement.appendChild(createTextElement(doc, "Utca_házszám", utcaHazszam));
        vasarloElement.appendChild(cimElement);

        for (String tel : telefon) {
            vasarloElement.appendChild(createTextElement(doc, "Telefonszám", tel));
        }

        return vasarloElement;
    }

    private static Element createBankkartyaElement(Document doc, String bid, String bank, String
kartyaszam, String lejaratiDatum) {
        Element bankkartyaElement = doc.createElement("Bankkártya");
        bankkartyaElement.setAttribute("BID", bid);

        bankkartyaElement.appendChild(createTextElement(doc, "Bank", bank));
        bankkartyaElement.appendChild(createTextElement(doc, "Kártyaszám", kartyaszam));
        bankkartyaElement.appendChild(createTextElement(doc, "Lejárati_dátum", lejaratiDatum));

        return bankkartyaElement;
```

```
    }

    private static Element createSzamitogepElement(Document doc, String szid, String nev, String ar,
String darab) {
        Element szamitogepElement = doc.createElement("Számítógép");
        szamitogepElement.setAttribute("SZID", szid);

        szamitogepElement.appendChild(createTextElement(doc, "Név", nev));
        szamitogepElement.appendChild(createTextElement(doc, "Ár", ar));
        szamitogepElement.appendChild(createTextElement(doc, "Darab", darab));

        return szamitogepElement;
    }

    private static Element createAlkatreszElement(Document doc, String aid, String nev, String ar,
String tipus) {
        Element alkatreszElement = doc.createElement("Alkatrész");
        alkatreszElement.setAttribute("AID", aid);

        alkatreszElement.appendChild(createTextElement(doc, "Név", nev));
        alkatreszElement.appendChild(createTextElement(doc, "Ár", ar));
        alkatreszElement.appendChild(createTextElement(doc, "Tipus", tipus));

        return alkatreszElement;
    }

    private static Element createFizetesElement(Document doc, String fiok, String bankkartya, String
jovahagyas) {
        Element fizetesElement = doc.createElement("Fizetés");
        fizetesElement.setAttribute("fiók", fiok);
        fizetesElement.setAttribute("bankkártya", bankkartya);
```

```
        fizetesElement.appendChild(createTextElement(doc, "Jóváhagyás", jovahagyas));

        return fizetesElement;
    }

    private static Element createVasarlasElement(Document doc, String fiok, String szamitogep, String
datum) {
        Element vasarlasElement = doc.createElement("Vásárlás");
        vasarlasElement.setAttribute("fiók", fiok);
        vasarlasElement.setAttribute("számítógép", szamitogep);

        vasarlasElement.appendChild(createTextElement(doc, "Dátum", datum));

        return vasarlasElement;
    }

    private static Element createOsszeszerelésElement(Document doc, String szamitogep, String
alkatresz, String darab) {
        Element osszeszerelésElement = doc.createElement("Összeszerelés");
        osszeszerelésElement.setAttribute("számítógép", szamitogep);
        osszeszerelésElement.setAttribute("alkatrész", alkatresz);

        osszeszerelésElement.appendChild(createTextElement(doc, "Darab", darab));

        return osszeszerelésElement;
    }

    private static Element createTextElement(Document doc, String tagName, String textContent) {
        Element element = doc.createElement(tagName);
        element.setTextContent(textContent);
        return element;
    }
```

```
// Az XML fájlba strukturált formában való írása
private 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);
}

// 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("/>");
}
```

```
    }
}
```