

# **Jegyzőkönyv**

**Adatkezelés XML környezetben**

**Cukrászda**

**Féléves feladat**

Készítette: **Gulyás Gábor**  
Neptunkód: **BFHRGP**

## Tartalom

1.	A feladat leírása, bevezetés: .....	3
2.	Az ER modell egyedei és tulajdonságai:.....	3
3.	Egyedek közötti kapcsolat: .....	4
4.	Az adatbázis ER-modell.....	5
5.	Az adatbázis konvertálása XDM modellre .....	6
6.	Az XDM modell alapján XML dokumentum készítése .....	6
7.	Az XML dokumentum alapján XMLSchema készítése .....	8
8.	DOM program - Adatolvasás.....	11
9.	Adatmódosítás .....	16
10.	Lekérdezések megvalósítása .....	19
11.	Adatírás.....	24

## 1. A feladat leírása, bevezetés:

A beadandó témája egy olyan adatbázis, amely Magyarország egyes cukrászdáit kezeli. Az adatbázisban megtalálhatók egyes cukrászdák, árult termékei. Lehetőség van termékeket online rendelni kiszállítással és fizetéssel, emiatt a vevő adatait is szükséges tárolni. Emellett a cukrászdában dolgozók adatait is megtalálhatjuk az adathalmazban.

## 2. Az ER modell egyedei és tulajdonságai:

### ▪ A cukrászda egyed tulajdonságai:

- CukrászdaID: A cukrászda egyed elsődleges kulcsa.
- Név: Egyes cukrászdák megnevezései.
- Nyitvatartás: A cukrászdák nyitvatartási ideje.
- Elérhetőség: Üzletek elérhetőségei.

### ▪ A termék egyed tulajdonságai:

- TermékID: A termék egyed elsődleges kulcsa
- Név: A termék neve
- Típus: A termék típusa.
- Egység ár
- CukrászdaID: Cukrászda egyed elsődleges kulcsa idegen kulcsként

### ▪ A vevő egyed tulajdonságai

- VevőID: A vevő egyed elsődleges kulcsa.
- Név: A vevő neve.
- Cím: A vevő címe. Összetett tulajdonság.
- Telefonszám: A vevő telefonszáma.

### ▪ A futár egyed tulajdonságai

- FutárID: A futár egyed elsődleges kulcsa.
- Név: A futár neve.
- Telefonszám: A futár telefonszáma.
- CukrászdaID: Cukrászda egyed elsődleges kulcsa idegen kulcsként

- **A kártya egyed tulajdonságai:**

- Kártyaszám: A kártya egyed elsődleges kulcsa.
- Típus: A kártya típusa.
- Lejárat dátum: A kártya lejárat dátuma.
- Bank: A bank neve
- VevőID: Vevő egyed elsődleges kulcsa idegen kulcsként

### **3. Egyedek közötti kapcsolat:**

- **Cukrászda és Termék:**

A cukrászda és termék egyedek között 1:N kapcsolat, mivel egy termék csak egy cukrászdához tartozhat, de egy cukrászdának lehet több terméke is.

- **Termék és Vevő:**

A termék és vevő egyedek között N:M kapcsolat van, mivel egy vevő rendelhet többfajta terméket is, és egy termékekből rendelhet több vevő is. Ennek az N:M kapcsolatnak vannak tulajdonságai (Rendelés: RendelésID, Mennyiség, Fizetendő összeg, TermékID, VevőID)

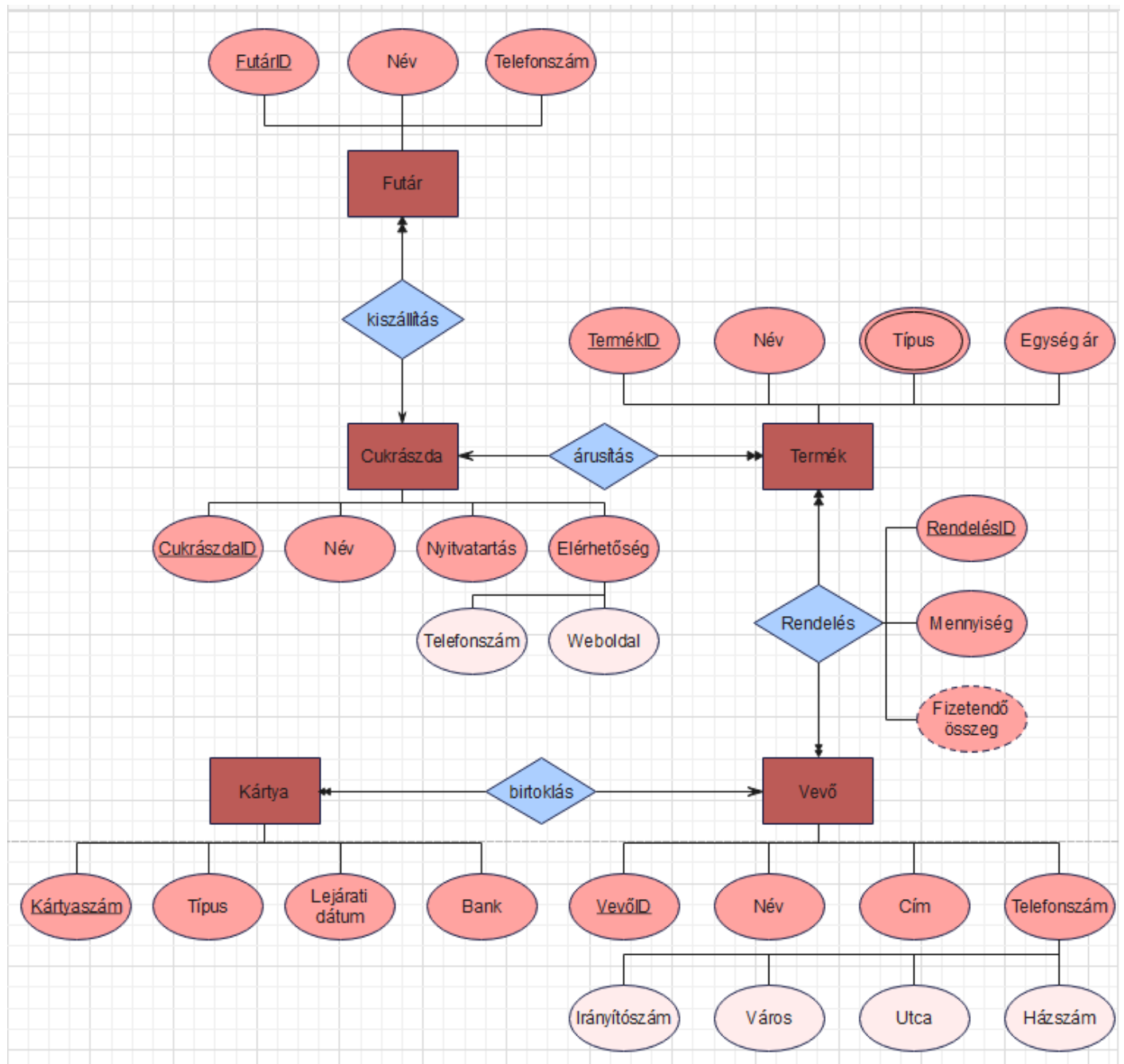
- **Cukrászda és Munkavállaló:**

A cukrászda és munkavállaló egyedek között 1:N kapcsolat van, mivel egy cukrászda alkalmazhat több munkavállalót, de egy munkavállaló csak egy cukrászdánál dolgozhat (rögzítették a munka szerződésben).

- **Vevő és Kártya:**

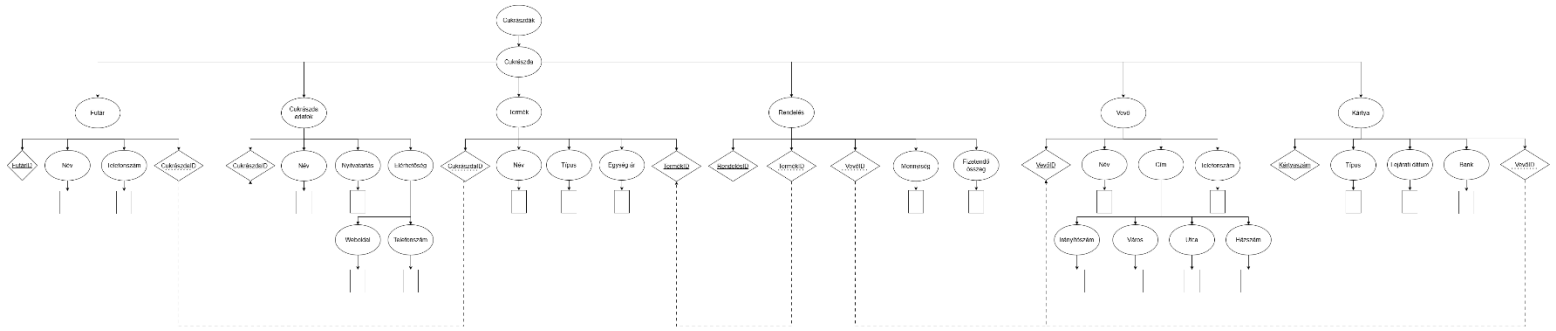
A vevő és kártya egyedek között 1:N kapcsolat van, mivel egy vevőnek lehet több kártyája is, de egy bankkártyának csak egy tulajdonosa lehet. (Ha azt feltételezzük, hogy egy embernek egy bankkártyája van akkor 1:1 kapcsolatról van szó, de most nem úgy értelmezzük)

#### 4. Az adatbázis ER-modell



1.ábra: Cukrászda ER modell

## 5. Az adatbázis konvertálása XDM modellre



2.ábra: Cukraszda XDM modell

## 6. Az XDM modell alapján XML dokumentum készítése

```
<?xml version="1.0" encoding="UTF-8"?>
<cukraszda xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="XMLSchemaBFHRGP.xsd">
  <cukraszda>
    <cukraszda_adatok CukraszdaID="1">
      <nev>Stühmer cukraszda</nev>
      <nyitvatartas>10:00-17:00</nyitvatartas>
      <weboldal>stuhmer.hu</weboldal>
      <telefonszam>0636517372</telefonszam>
    </cukraszda_adatok>

    <futar FutarID="1">
      <nev>Kis Béla</nev>
      <telefonszam>06301234567</telefonszam>
      <CukraszdaID>1</CukraszdaID>
    </futar>
    <futar FutarID="2">
      <nev>Nagy Béla</nev>
      <telefonszam>06707654321</telefonszam>
      <CukraszdaID>1</CukraszdaID>
    </futar>

    <termek TermekID="1">
      <nev>Melódia</nev>
      <tipus>szelet</tipus>
      <egyseg_ar>500</egyseg_ar>
      <CukraszdaID>1</CukraszdaID>
    </termek>
    <termek TermekID="2">
      <nev>Korfu</nev>
      <tipus>szelet</tipus>
      <egyseg_ar>450</egyseg_ar>
      <CukraszdaID>1</CukraszdaID>
    </termek>
    <termek TermekID="3">
      <nev>Sós-mogyorós barack zsúr</nev>
      <tipus>torta</tipus>
      <egyseg_ar>4500</egyseg_ar>
      <CukraszdaID>1</CukraszdaID>
    </termek>
  </cukraszda>
</cukraszda>
```

```
<vevo VevoID="1">
  <nev>Nagy Béla</nev>
  <iranyitoszam>3300</iranyitoszam>
  <varos>Eger</varos>
  <utca>Merengő</utca>
  <hazszam>1</hazszam>
  <telefonszam>06309876532</telefonszam>
</vevo>
<vevo VevoID="2">
  <nev>Nagy János</nev>
  <iranyitoszam>3535</iranyitoszam>
  <varos>Miskolc</varos>
  <utca>Eper</utca>
  <hazszam>2</hazszam>
  <telefonszam>06608884441</telefonszam>
</vevo>

<rendeles RendelesID="1">
  <mennyiseg>2</mennyiseg>
  <fizetendo_osszeg>1000</fizetendo_osszeg>
  <TermekID>1</TermekID>
  <VevoID>1</VevoID>
</rendeles>

<rendeles RendelesID="2">
  <mennyiseg>1</mennyiseg>
  <fizetendo_osszeg>4500</fizetendo_osszeg>
  <TermekID>3</TermekID>
  <VevoID>2</VevoID>
</rendeles>

<rendeles RendelesID="3">
  <mennyiseg>3</mennyiseg>
  <fizetendo_osszeg>1350</fizetendo_osszeg>
  <TermekID>2</TermekID>
  <VevoID>1</VevoID>
</rendeles>

<kartya Kartyaszam="1177339100111222">
  <tipus>SZÉP</tipus>
  <lejarati_datum>2022-11-30</lejarati_datum>
  <bank>OTP</bank>
  <VevoID>1</VevoID>
</kartya>
<kartya Kartyaszam="117733901234567">
  <tipus>Bank</tipus>
  <lejarati_datum>2023-05-02</lejarati_datum>
  <bank>MKB</bank>
  <VevoID>1</VevoID>
</kartya>
<kartya Kartyaszam="8823569123547632">
  <tipus>SZÉP</tipus>
  <lejarati_datum>2021-12-31</lejarati_datum>
  <bank>KH</bank>
  <VevoID>2</VevoID>
</kartya>
</cukraszda>
</cukraszdek>
```

## 7. Az XML dokumentum alapján XMLSchema készítése

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="cukraszda">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="cukraszda">
          <xs:complexType>
            <xs:sequence>
<xs:element name="cukraszda_adatok" type="cukraszdaTipus" maxOccurs="unbounded" minOccurs="0"
/>
<xs:element name="futar" type="futarTipus" maxOccurs="unbounded" minOccurs="0" />
<xs:element name="termek" type="termekTipus" maxOccurs="unbounded" minOccurs="0" />
<xs:element name="vevo" type="vevoTipus" maxOccurs="unbounded" minOccurs="0" />
<xs:element name="rendeles" type="rendelesTipus" maxOccurs="unbounded" minOccurs="0" />
<xs:element name="kartya" type="kartyaTipus" maxOccurs="unbounded" minOccurs="0" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>

    <xs:key name="CukraszdaID">
      <xs:selector xpath="cukraszda/cukraszda_adatok"/>
      <xs:field xpath="@CukraszdaID"/>
    </xs:key>

    <xs:key name="FutarID">
      <xs:selector xpath="cukraszda/futar"/>
      <xs:field xpath="@FutarID"/>
    </xs:key>

    <xs:key name="TermekID">
      <xs:selector xpath="cukraszda/termek"/>
      <xs:field xpath="@TermekID"/>
    </xs:key>

    <xs:key name="VevoID">
      <xs:selector xpath="cukraszda/vevo"/>
      <xs:field xpath="@VevoID"/>
    </xs:key>

    <xs:key name="RendelesID">
      <xs:selector xpath="cukraszda/rendeles"/>
      <xs:field xpath="@RendelesID"/>
    </xs:key>

    <xs:key name="Kartyaszam">
      <xs:selector xpath="cukraszda/kartya"/>
      <xs:field xpath="@Kartyaszam"/>
    </xs:key>

    <xs:keyref name="cukraszda-futar" refer="CukraszdaID">
      <xs:selector xpath="cukraszda/futar/CukraszdaID"/>
      <xs:field xpath="."/>
    </xs:keyref>

    <xs:keyref name="cukraszda-termek" refer="CukraszdaID">
      <xs:selector xpath="cukraszda/termek/CukraszdaID"/>
      <xs:field xpath="."/>
    </xs:keyref>
  </xs:element>
</xs:schema>
```



```

<xs:keyref name="termek-rendeles" refer="TermekID">
  <xs:selector xpath="cukraszda/rendeles/TermekID"/>
  <xs:field xpath="."/>
</xs:keyref>

<xs:keyref name="vevo-rendeles" refer="VevoID">
  <xs:selector xpath="cukraszda/rendeles/VevoID"/>
  <xs:field xpath="."/>
</xs:keyref>

<xs:keyref name="vevo-kartya" refer="VevoID">
  <xs:selector xpath="cukraszda/kartya/VevoID"/>
  <xs:field xpath="."/>
</xs:keyref>
</xs:element>

<xs:complexType name="cukraszdaTipus">
  <xs:sequence>
    <xs:element type="xs:string" name="nev" />
    <xs:element type="xs:string" name="nyitvatartas" />
    <xs:element type="xs:string" name="weboldal"/>
    <xs:element type="xs:int" name="telefonszam"/>
  </xs:sequence>
  <xs:attribute type="xs:short" name="CukraszdaID" use="required" />
</xs:complexType>

<xs:complexType name="futarTipus">
  <xs:sequence>
    <xs:element type="xs:string" name="nev" />
    <xs:element type="xs:string" name="telefonszam" />
    <xs:element type="xs:short" name="CukraszdaID" />
  </xs:sequence>
  <xs:attribute type="xs:short" name="FutarID" use="required" />
</xs:complexType>

<xs:complexType name="termekTipus">
  <xs:sequence>
    <xs:element type="xs:string" name="nev" />
    <xs:element type="xs:string" name="tipus" />
    <xs:element type="xs:int" name="egyseg_ar" />
    <xs:element type="xs:short" name="CukraszdaID" />
  </xs:sequence>
  <xs:attribute type="xs:short" name="TermekID" use="required" />
</xs:complexType>

<xs:complexType name="vevoTipus">
  <xs:sequence>
    <xs:element type="xs:string" name="nev" />
    <xs:element type="xs:short" name="iranyitoszam" />
    <xs:element type="xs:string" name="varos" />
    <xs:element type="xs:string" name="utca" />
    <xs:element type="xs:short" name="hazszam" />
    <xs:element type="xs:long" name="telefonszam" />
  </xs:sequence>
  <xs:attribute type="xs:short" name="VevoID" use="required" />
</xs:complexType>

```

```
<xs:complexType name="rendelesTipus">
  <xs:sequence>
    <xs:element type="xs:int" name="mennyiseg" />
    <xs:element type="xs:int" name="fizetendo_osszeg" />
    <xs:element type="xs:short" name="TermekID" />
    <xs:element type="xs:short" name="VevoID" />
  </xs:sequence>
  <xs:attribute type="xs:short" name="RendelesID" use="required" />
</xs:complexType>

<xs:complexType name="kartyaTipus">
  <xs:sequence>
    <xs:element type="xs:string" name="tipus" />
    <xs:element type="xs:date" name="lejarati_datum" />
    <xs:element type="xs:string" name="bank" />
    <xs:element type="xs:short" name="VevoID" />
  </xs:sequence>
  <xs:attribute type="xs:long" name="Kartyaszam" use="required" />
</xs:complexType>
</xs:schema>
```

## 8. DOM program - Adatolvasás

```
package hu.domparse.bfhrgp;

import java.io.File;
import java.io.IOException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DOMReadBFHRGP {

    public static void main(String[] args){
        try {
            //Fajl betoltes
            File file = new File("XMLBFHRGP.xml");

            //Dokumentum olvaso letrehozasa
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            dbf.setValidating(true);
            DocumentBuilder db = dbf.newDocumentBuilder();
            db.setErrorHandler(new hibakezeles());

            //Dokumentum letrehozasa fajlbol
            Document doc = db.parse(file);

            doc.getDocumentElement().normalize();

            //Gyokerelem lekerdezés
```

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

```
//Beolvas: Cukraszda elem attributumai es alelemei
NodeList nodeList = doc.getElementsByTagName("cukraszda_adatok");
for (int i = 0; i < nodeList.getLength(); i++)
{
    Node node = nodeList.item(i);
    System.out.println("\n" + node.getNodeName() + " " + (i + 1));
    if (node.getNodeType() == Node.ELEMENT_NODE)
    {
        Element Elem = (Element) node;
        System.out.println("CukraszdaID: "+
Elem.getAttribute("CukraszdaID"));
        System.out.println("Nev: "+
Elem.getElementsByTagName("nev").item(0).getTextContent());
        System.out.println("Nyitvatartas: "+
Elem.getElementsByTagName("nyitvatartas").item(0).getTextContent());
        System.out.println("Weboldal: "+
Elem.getElementsByTagName("weboldal").item(0).getTextContent());
        System.out.println("Telefonszam: "+
Elem.getElementsByTagName("telefonszam").item(0).getTextContent());
    }
}
```

```
//Beolvas: Futar elem attributumai es alelemei
nodeList = doc.getElementsByTagName("futar");
for (int i = 0; i < nodeList.getLength(); i++)
{
    Node node = nodeList.item(i);
    System.out.println("\n" + node.getNodeName() + " " + (i + 1));
    if (node.getNodeType() == Node.ELEMENT_NODE)
    {
        Element Elem = (Element) node;
        System.out.println("FutarID: "+ Elem.getAttribute("FutarID"));
        System.out.println("Nev: "+
Elem.getElementsByTagName("nev").item(0).getTextContent());
    }
}
```

```

        System.out.println("Telefonszam: "+
Elem.getElementsByTagName("telefonszam").item(0).getTextContent());

        System.out.println("CukraszdaID: "+
Elem.getElementsByTagName("CukraszdaID").item(0).getTextContent());
    }
}

//Beolvas: Termek elem attributumai es alelemei
nodeList = doc.getElementsByTagName("termek");
for (int i = 0; i < nodeList.getLength(); i++)
{
    Node node = nodeList.item(i);
    System.out.println("\n" + node.getNodeName() + " " + (i + 1));
    if (node.getNodeType() == Node.ELEMENT_NODE)
    {
        Element Elem = (Element) node;
        System.out.println("TermekID: "+
Elem.getAttribute("TermekID"));
        System.out.println("Nev: "+
Elem.getElementsByTagName("nev").item(0).getTextContent());
        System.out.println("Tipus: "+
Elem.getElementsByTagName("tipus").item(0).getTextContent());
        System.out.println("Egyseg ar: "+
Elem.getElementsByTagName("egyseg_ar").item(0).getTextContent());
        System.out.println("CukraszdaID: "+
Elem.getElementsByTagName("CukraszdaID").item(0).getTextContent());
    }
}

//Beolvas: Vevo elem attributumai es alelemei
nodeList = doc.getElementsByTagName("vevo");
for (int i = 0; i < nodeList.getLength(); i++)
{
    Node node = nodeList.item(i);
    System.out.println("\n" + node.getNodeName() + " " + (i + 1));
    if (node.getNodeType() == Node.ELEMENT_NODE)
    {

```

```

        Element Elem = (Element) node;

        System.out.println("VevoID: " + Elem.getAttribute("VevoID"));

        System.out.println("Nev: " +
Elem.getElementsByTagName("nev").item(0).getTextContent());

        System.out.println("Irányítószám: " +
Elem.getElementsByTagName("iranyitoszam").item(0).getTextContent());

        System.out.println("Varos: " +
Elem.getElementsByTagName("varos").item(0).getTextContent());

        System.out.println("Utca: " +
Elem.getElementsByTagName("utca").item(0).getTextContent());

        System.out.println("Házzszám: " +
Elem.getElementsByTagName("hazsszam").item(0).getTextContent());

        System.out.println("Telefonszám: " +
Elem.getElementsByTagName("telefonszam").item(0).getTextContent());

    }

}

```

```

//Beolvas: Rendeles elem attributumai es alelemei
nodeList = doc.getElementsByTagName("rendeles");
for (int i = 0; i < nodeList.getLength(); i++)
{
    Node node = nodeList.item(i);

    System.out.println("\n" + node.getNodeName() + " " + (i + 1));

    if (node.getNodeType() == Node.ELEMENT_NODE)
    {
        Element Elem = (Element) node;

        System.out.println("RendelesID: " +
Elem.getAttribute("RendelesID"));

        System.out.println("Mennyiség: " +
Elem.getElementsByTagName("mennyiseg").item(0).getTextContent());

        System.out.println("Fizetendo osszeg: " +
Elem.getElementsByTagName("fizetendo_osszeg").item(0).getTextContent());

        System.out.println("TermekID: " +
Elem.getElementsByTagName("TermekID").item(0).getTextContent());

        System.out.println("VevoID: " +
Elem.getElementsByTagName("VevoID").item(0).getTextContent());

    }

}

```

```

//Beolvas: Kartya elem attributumai es alelemei
nodeList = doc.getElementsByTagName("kartya");
for (int i = 0; i < nodeList.getLength(); i++)
{
    Node node = nodeList.item(i);
    System.out.println("\n" + node.getNodeName() + " " + (i + 1));
    if (node.getNodeType() == Node.ELEMENT_NODE)
    {
        Element Elem = (Element) node;
        System.out.println("Kartyaszam: "+
Elem.getAttribute("Kartyaszam"));
        System.out.println("Tipus: "+
Elem.getElementsByTagName("tipus").item(0).getTextContent());
        System.out.println("Lejarati datum: "+
Elem.getElementsByTagName("lejarati_datum").item(0).getTextContent());
        System.out.println("Bank: "+
Elem.getElementsByTagName("bank").item(0).getTextContent());
        System.out.println("VevoID: "+
Elem.getElementsByTagName("VevoID").item(0).getTextContent());

    }
}

catch (ParserConfigurationException pce) {pce.printStackTrace();}
catch(SAXException se) { }
catch(IOException ioe) { }
}
}

```

## 9. DOM program - Adatmódosítás

```
package hu.domparse.bfhrgp;

import java.io.File;
import java.io.IOException;
import java.util.Scanner;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DOMModifyBFHRGP {

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

        //Fajl betoltes
        File file = new File("XMLBFHRGP.xml");

        //Dokumentum olvaso letrehozasa
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        dbf.setValidating(true);
        DocumentBuilder db = dbf.newDocumentBuilder();
        db.setErrorHandler(new hibakezeles());

        //Dokumentum letrehozasa fajlbol
        Document doc = db.parse(file);

        doc.getDocumentElement().normalize();

        modositVevo(doc);
    }

    //Uj fajl letrehozasa a modosított adatokkal
    public static void modosítottxml(Document doc) throws TransformerException {
        TransformerFactory transformerFactory =
        TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new
        File("modosított_XMLBFHRGP.xml"));
        transformer.transform(source, result);
    }

    private static void modositVevo(Document doc) throws TransformerException {
        //Lekerjuk a vevo elembe tarolt adatokat
    }
}
```



```

        NodeList nodeList = doc.getElementsByTagName("vevo");
        for (int j = 0; j < nodeList.getLength(); j++)
        {
            Node node = nodeList.item(j);
            System.out.println("\n" + node.getNodeName() + " " + (j + 1));
            if (node.getNodeType() == Node.ELEMENT_NODE)
            {
                Element Elem = (Element) node;
                System.out.println("VevoID: " + Elem.getAttribute("VevoID"));
                System.out.println("Nev: " +
Elem.getElementsByTagName("nev").item(0).getTextContent());
                System.out.println("Irányítószám: " +
Elem.getElementsByTagName("iranyitoszam").item(0).getTextContent());
                System.out.println("Varos: " +
Elem.getElementsByTagName("varos").item(0).getTextContent());
                System.out.println("Utca: " +
Elem.getElementsByTagName("utca").item(0).getTextContent());
                System.out.println("Házzszám: " +
Elem.getElementsByTagName("hazszam").item(0).getTextContent());
                System.out.println("Telefonszám: " +
Elem.getElementsByTagName("telefonszam").item(0).getTextContent());
            }
        }

        //Vevok azonositojanak bekereése
        System.out.println("\nUdvazoljuk a vevoi ügyfeliszolgalaton!\nAdja
meg melyik vevo adatait szeretne modositani!");
        //Bekerjuk a vevo id-t aminek az adatait modositjuk
        Scanner sc = new Scanner(System.in);
        System.out.print("\nid:");
        String id = sc.nextLine();
        // Bekerjuk az uj adatokat
        System.out.print("Nev: ");
        String nev = sc.nextLine();
        System.out.print("Irányítószám: ");
        String iranyitoszam = sc.nextLine();
        System.out.print("Varos: ");
        String varos = sc.nextLine();
        System.out.print("Utca: ");
        String utca = sc.nextLine();
        System.out.print("Házzszám: ");
        String hazszam = sc.nextLine();
        System.out.print("Telefonszám: ");
        String telefonszam = sc.nextLine();
        sc.close();
        //Lekerdezzek az elemeket, majd setTextContent-el modositjuk
        NodeList elemLista = doc.getElementsByTagName("vevo");
        for (int i = 0; i < elemLista.getLength(); i++) {
            Node nNode = elemLista.item(i);
            if (nNode.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) nNode;
                String sid = element.getAttribute("VevoID");
                if (sid.equals(id)) {
                    Node node1 =
element.getElementsByTagName("nev").item(0);
                    node1.setTextContent(nev);
                    Node node2 =
element.getElementsByTagName("iranyitoszam").item(0);
                    node2.setTextContent(iranyitoszam);
                    Node node3 =
element.getElementsByTagName("varos").item(0);
                    node3.setTextContent(varos);

```

```

        Node node4 =
element.getElementsByTagName("utca").item(0);
        node4.setTextContent(utca);
        Node node5 =
element.getElementsByTagName("hazszam").item(0);
        node5.setTextContent(hazszam);
        Node node6 =
element.getElementsByTagName("telefonszam").item(0);
        node6.setTextContent(telefonszam);
        System.out.println("Sikeres modositas");
    }
}
}
modositottxml(doc); //Letrehozzuk a modositott_XMLBFHRGP-t
}
}

```

## 10.DOM program - Lekérdezések megvalósítása

```
package hu.domparse.bfhrgp;

import java.io.File;
import java.io.IOException;

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

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DOMQueryBFHRGP {

    public static void main(String[] args){
        try {
            //Fajl betoltes
            File file = new File("XMLBFHRGP.xml");

            //Dokumentum olvaso letrehozasa
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            dbf.setValidating(true);
            DocumentBuilder db = dbf.newDocumentBuilder();
            db.setErrorHandler(new hibakezeles());

            //Dokumentum letrehozasa fajlbol
            Document doc = db.parse(file);

            doc.getDocumentElement().normalize();
```

```

//1. lekerdezes: 2-es azonositoju termek kiirasa
String TermekID = "2";
NodeList termekList = doc.getElementsByTagName("termek");
for (int i = 0; i < termekList.getLength(); i++) {
    Element termek = (Element) termekList.item(i);
    String termekIdAttribute = termek.getAttribute("TermekID");
    if (termekIdAttribute.equals(TermekID)) {
        String nev =
termek.getElementsByTagName("nev").item(0).getTextContent();

        String tipus =
termek.getElementsByTagName("tipus").item(0).getTextContent();

        String ar =
termek.getElementsByTagName("egyseg_ar").item(0).getTextContent();

        System.out.println("1. lekérdezés: 2-es azonosítóju termék
kiirasa");

        System.out.println("A " + TermekID + " ID-jú termékneve: " +
nev);

        System.out.println("Tipusa: " + tipus);
        System.out.println("Ára: " + ar);
        break;
    }
}

//2. lekerdezes: legdragabb termek kiirasa
NodeList legdragabbtermekList = doc.getElementsByTagName("termek");

String legdragabbNev = "";
String legdragabbTipus = "";
int legdragabbAr = 0;

for (int i = 0; i < legdragabbtermekList.getLength(); i++) {
    Element termek = (Element) legdragabbtermekList.item(i);

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

    String tipus =
termek.getElementsByTagName("tipus").item(0).getTextContent().trim();

```

```

        int ar =
Integer.parseInt(termek.getElementsByTagName("egyseg_ar").item(0).getTextContent().
trim());

        // i-edik termék ára dragabb-e mint a legdragabb
        if (ar > legdragabbAr) {
            legdragabbAr = ar;
            legdragabbNev = nev;
            legdragabbTipus = tipus;
        }
    }

    System.out.println("\n2. lekérdezés: legdragabb termék kiirasa");
    System.out.println("Legdrágább termék:");
    System.out.println("  Név: " + legdragabbNev);
    System.out.println("  Típus: " + legdragabbTipus);
    System.out.println("  Ár: " + legdragabbAr);

    // 3 lekerdezes: 1200 ft feletti rendelest leado vevok kiirasa
    NodeList vevoList = doc.getElementsByTagName("vevo");
    NodeList rendelesList = doc.getElementsByTagName("rendeles");

    for (int i = 0; i < vevoList.getLength(); i++) {
        Element vevo = (Element) vevoList.item(i);

        String vevoNev =
vevo.getElementsByTagName("nev").item(0).getTextContent().trim();
        String vevoID = vevo.getAttribute("VevoID");

        // ellenorzi hogy van-e 1200 Ft fölötti rendelése
        boolean vanrendeles = false;

        for (int j = 0; j < rendelesList.getLength(); j++) {

```

```

        Element rendeles = (Element) rendelesList.item(j);

        String rendelesVevoID =
rendeles.getElementsByTagName("VevoID").item(0).getTextContent().trim();

        if (rendelesVevoID.equals(vevoID)) {

            String ar =
rendeles.getElementsByTagName("fizetendo_osszeg").item(0).getTextContent().trim();
            if (Integer.parseInt(ar) > 1200) {
                vanrendeles = true;
                break;
            }
        }
    }

    // Csak azon vevok kerulnek kiirasra, akiknek 1200 Ft folotti a
rendelesuk

    if (vanrendeles) {

        System.out.println("\n3. lekérdezés: 1200 ft feletti rendelest
leado vevok kiirasa");

        System.out.println("Vevő neve: " + vevoNev);

        for (int j = 0; j < rendelesList.getLength(); j++) {

            Element rendeles = (Element) rendelesList.item(j);

            String rendelesVevoID =
rendeles.getElementsByTagName("VevoID").item(0).getTextContent().trim();

            if (rendelesVevoID.equals(vevoID)) {

                String termékID =
rendeles.getElementsByTagName("TermekID").item(0).getTextContent().trim();

                String mennyiseg =
rendeles.getElementsByTagName("mennyiseg").item(0).getTextContent().trim();

                String fizetendo =
rendeles.getElementsByTagName("fizetendo_osszeg").item(0).getTextContent().trim();

                System.out.println("    Rendelés:");

                System.out.println("        Termék ID: " + termékID);

                System.out.println("        Mennyiség: " + mennyiseg);

                System.out.println("        Fizetendő összeg: " +
fizetendo);

```

```
        }  
    }  
}  
}
```

```
catch (ParserConfigurationException pce) {pce.printStackTrace();}  
catch(SAXException se) { }  
catch(IOException ioe) { }  
}  
}
```

## 11.DOM program - Adatírás

```
package hu.domparse.bfhrgp;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

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

public class DOMWriteBFHRGP {

    public static void main(String[] args) throws Exception {

        new File("XMLBFHRGP.xml");
        // XML beolvasas
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document doc = builder.parse("XMLBFHRGP.xml");

        writeNode(doc.getDocumentElement(), 0);
        fajlkiiras(doc);

    }
```



```

public static void writeNode(Node node, int indent) {
    // sor behuzas
    for (int i = 0; i < indent; i++) {
        System.out.print(" ");
    }

    // elemek es attributumok kiirasa
    if (node.getNodeType() == Node.ELEMENT_NODE) {
        Element element = (Element) node;
        System.out.print("<" + element.getNodeName());

        NamedNodeMap attributes = element.getAttributes();
        for (int i = 0; i < attributes.getLength(); i++) {
            Node attribute = attributes.item(i);
            System.out.print(" " + attribute.getNodeName() + "=\"" +
attribute.getNodeValue() + "\"");
        }
        System.out.println(">");
    } else if (node.getNodeType() == Node.TEXT_NODE) {
        String textContent = node.getTextContent().trim();
        if (!textContent.isEmpty()) {
            System.out.println(textContent);
        }
    }

    // gyerek elemek feldolgozasa rekurziv modon
    NodeList children = node.getChildNodes();
    for (int i = 0; i < children.getLength(); i++) {
        writeNode(children.item(i), indent + 1);
    }

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        for (int i = 0; i < indent; i++) {

```

```

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

//Uj fajl létrehozása
public static void fajlkiiras(Document doc) throws TransformerException
{
    TransformerFactory transformerFactory =
TransformerFactory.newInstance();

    Transformer transformer = transformerFactory.newTransformer();
    DOMSource source = new DOMSource(doc);
    StreamResult result = new StreamResult(new
File("XMLBFHRGP1.xml"));
    transformer.transform(source, result);
    System.out.println("\nXMLBFHRGP1.xml sikeresen létre lett
hozva");
}
}

```