JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Ablakgyártó cég nyilvántartó rendszere

Készítette: **Bakos Dominik Dávid**

Neptunkód: **M95ETT**

Dátum: **2022. 11. 26**

Tartalom

1) A Feladat leírása

A feladatban célom egy ablakgyártó cég modelljének az elkészítése volt, melyben követhetjük a megrendelések státuszát. Ez egy általános, leegyszerűsített modell. A valóságban ennél összetetteb struktúrával rendelkeznek a cégek.

A **rendelések**ben a megrendelés dátuma, és a megrendelő néhány adata található (a kapcsolatfelvétel dátuma, a megrendelő neve és a település). Ezen adatok alapján egyértelműen beazonosítható a megrendelés.

Minden rendeléshez tartozik egy-egy **üveg** típus. Ez a megrendeléshez tartozó nyílászárókba kerülő üvegek adatait tartalmazza. A vásárló választhat három- és ötrétegű üvegek közül, kérhet hővédő fóliát az üvegekre. Egy rendeléshez egy elemben rögzítjük az összes szükséges üveget, ezért szerepel idegen kulcsként a rendelésszám is.

A nyílászárók elkészítéséhez szükséges **profil**ok használata is. Az üvegekhez hasonlóan egy rendeléshez egyetlen profil elem csatlakozhat, ezért idegen kulcsként bevezetésre került a rendelésszám. A profil osztályban műanyag, fa és alumínium anyagokat és a méretet találjuk. A standard magasság 90 cm, ezért csak a hossz kerül rögzítésre.

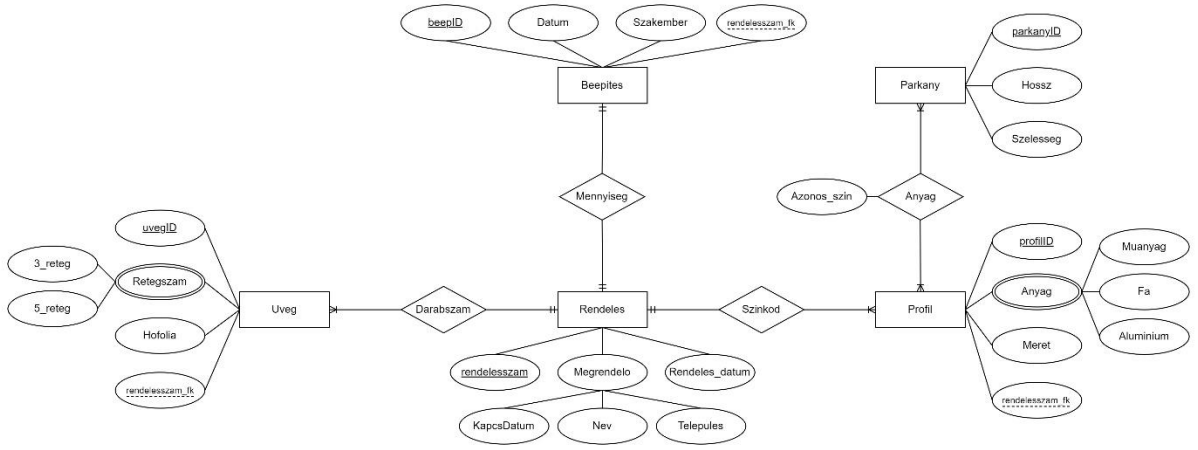
Opcionálisan **párkány**t is választhat a vásárló. Ebben az esetben a hossz és szélesség kerül tárolásra, mivel itt mindkettő az adott fal méretétől függ. Mivel egy méretű párkány több profilhoz is tartozhat, ezért a kapcsolatok egy külön **anyagok** nevű osztályban kerülnek tárolásra.

A cég biztosít **beépítés**i szolgáltatást is. Ez egy nyilvántartás, jelezve milyen dátummal, hány ember végezte a beépítést. A rendelésszám jelzi, mely ablakok kerültek beépítésre.

2) Első feladat

2.1) Az adatbázis ER modellje

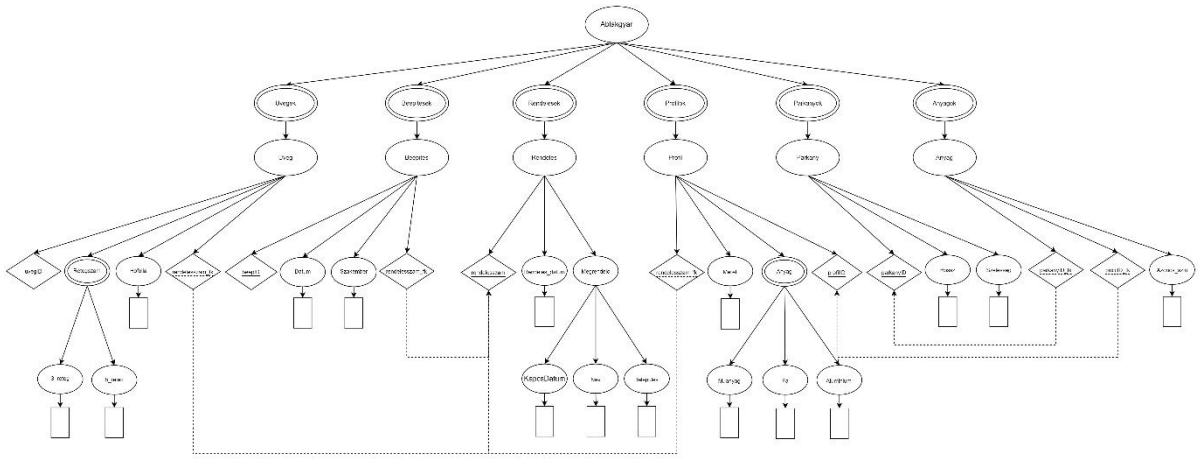
Az ER modell elkészítése során törekedtem az áltáthatóságra, és az esztétikumra. Az egyedek és tulajdonságok egyvonalba kerültek, amennyiben azonos szinten vannak. Az előírásoknak megfelelően jelöltem a tulajdonságokat. A tulajdonságok kialakítása során szerettem volna minél egyértelműbben és átláthatóbban lefedni a kívánt funkciót.



2.2) Az adatbázis konvertálása XDM modellre

Az XML dokumentumban egy-egy egyedből több példányt is szeretnék létrehozni. Ezért szükséges volt az összes egyedet egy szülő egyedbe. Az elnevezést a legkönnyebben a többes számú ragozással tudtam megoldani.

Az egyedek közötti kapcsolatok közül az Anyagból szükséges volt egy új osztály létrehozása, mivel a Profil és Párkány között több-több kapcsolat van. A két idegen kulcson túl még egy azonos szín tulajdonság került az osztályba. Ez azt jelzi, hogy a profil és a párkány színe megegyezik-e.



Az összes egyedet összefogó root elem az ablakgyár lett

2.3) Az XDM modell alapján XML dokumentum készítése

Az XDM modellt követve könnyedén létre tudtam hozni az XML kódot. Minden egyedből három példányt készítettem.

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

<Ablakgyar xsi:noNamespaceSchemaLocation="./XMLSchemaM95ETT.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <!-- Üvegek példányosítása -->
    <Uvegek>
        <Uveg uvegID="145">
            <Retegszam>
                <Haromreteg>2</Haromreteg>
                <Otreteg>1</Otreteg>
            </Retegszam>
            <Hofolia>3</Hofolia>
            <rendelesszam_fk>182</rendelesszam_fk>
        </Uveg>

        <Uveg uvegID="146">
            <Retegszam>
                <Haromreteg>0</Haromreteg>
                <Otreteg>4</Otreteg>
            </Retegszam>
            <Hofolia>2</Hofolia>
            <rendelesszam_fk>185</rendelesszam_fk>
        </Uveg>
```

```
    <Uveg uvegID="147">
        <Retegszam>
            <Haromreteg>5</Haromreteg>
            <Otreteg>5</Otreteg>
        </Retegszam>
        <Hofolia>3</Hofolia>
        <rendelesszam_fk>186</rendelesszam_fk>
    </Uveg>
</Uvegek>

<!-- Beépítések példányosítása -->
<Beepitesek>
    <Beepites beepID="034">
        <Datum>2022.06.15</Datum>
        <Szakember>2</Szakember>
        <rendelesszam_fk>182</rendelesszam_fk>
    </Beepites>

    <Beepites beepID="035">
        <Datum>2022.07.23</Datum>
        <Szakember>2</Szakember>
        <rendelesszam_fk>185</rendelesszam_fk>
    </Beepites>

    <Beepites beepID="036">
        <Datum>2022.08.10</Datum>
        <Szakember>4</Szakember>
        <rendelesszam_fk>186</rendelesszam_fk>
    </Beepites>
</Beepitesek>
<!-- Rendelések példányosítása -->
<Rendelesek>
    <Rendeles rendelesszam="182">
        <Rendeles_datum>2022.05.11</Rendeles_datum>
        <Megrendelo>
            <KapcsoaltfelvetelDatum>2022.05.12</KapcsoaltfelvetelDatum>
            <Nev>Horváth Pálné</Nev>
            <Telepules>Maklár</Telepules>
        </Megrendelo>
    </Rendeles>

    <Rendeles rendelesszam="185">
        <Rendeles_datum>2022.06.20</Rendeles_datum>
        <Megrendelo>
            <KapcsoaltfelvetelDatum>2022.04.27</KapcsoaltfelvetelDatum>
            <Nev>Kiss Béla</Nev>
            <Telepules>Miskolc</Telepules>
        </Megrendelo>
    </Rendeles>
```

```
    <Rendeles rendelesszam="186">
        <Rendeles_datum>2022.07.26</Rendeles_datum>
        <Megrendelo>
            <KapcsoaltfelvetelDatum>2022.06.02</KapcsoaltfelvetelDatum>
            <Nev>Nagy Lajos</Nev>
            <Telepules>Kisköre</Telepules>
        </Megrendelo>
    </Rendeles>
</Rendelesek>

<!-- Profilok példányosítása -->
<Profilok>
    <Profil profilID="003">
        <Meret>120</Meret>
        <Anyag>
            <Muanyag>3</Muanyag>
            <Fa>0</Fa>
            <Aluminium>0</Aluminium>
        </Anyag>
        <rendelesszam_fk>182</rendelesszam_fk>
    </Profil>

    <Profil profilID="008">
        <Meret>90</Meret>
        <Anyag>
            <Muanyag>2</Muanyag>
            <Fa>2</Fa>
            <Aluminium>0</Aluminium>
        </Anyag>
        <rendelesszam_fk>185</rendelesszam_fk>
    </Profil>
    <Profil profilID="012">
        <Meret>140</Meret>
        <Anyag>
            <Muanyag>5</Muanyag>
            <Fa>4</Fa>
            <Aluminium>1</Aluminium>
        </Anyag>
        <rendelesszam_fk>186</rendelesszam_fk>
    </Profil>
</Profilok>
```

```xml
    <!-- Párkányok példányosítása -->
    <Parkanyok>
        <Parkany parkanyID="005">
            <Hossz>120</Hossz>
            <Szelesseg>35</Szelesseg>
        </Parkany>

        <Parkany parkanyID="013">
            <Hossz>90</Hossz>
            <Szelesseg>50</Szelesseg>
        </Parkany>

        <Parkany parkanyID="017">
            <Hossz>140</Hossz>
            <Szelesseg>40</Szelesseg>
        </Parkany>
    </Parkanyok>

    <!-- Anyagok példányosítása -->
    <Anyagok>
        <Anyag>
            <parkanyID_fk>005</parkanyID_fk>
            <profilID_fk>003</profilID_fk>
            <AzonosSzin>Igen</AzonosSzin>
        </Anyag>

        <Anyag>
            <parkanyID_fk>013</parkanyID_fk>
            <profilID_fk>008</profilID_fk>
            <AzonosSzin>Nem</AzonosSzin>
        </Anyag>

        <Anyag>
            <parkanyID_fk>017</parkanyID_fk>
            <profilID_fk>012</profilID_fk>
            <AzonosSzin>Igen</AzonosSzin>
        </Anyag>
    </Anyagok>
</Ablakgyar>
```

2.4) Az XML dokumentum alapján XMLSchema készítése

A schema elkészítése során ügyeltem a szükséges mezők jelölésére. Az idegen kulcsok tartalma a jelölt elsődleges kulcsokra lett beállítva, az elsődleges kulcsok pedig csak három számból állhatnak. Minden egyedhez saját típust hoztam létre.

```xml
<?xml version="1.0" encoding="utf-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="qualified">
    <xs:element name="Ablakgyar">
        <xs:complexType>
            <xs:sequence>

                <xs:element name="Uvegek">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Uveg" type="UvegTipus" minOccurs="0" maxOccurs="unbounded" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>

                <xs:element name="Beepitesek">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Beepites" type="BeepitesTipus" minOccurs="0" maxOccurs="unbounded" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>

                <xs:element name="Rendelesek">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Rendeles" type="RendelesTipus" minOccurs="0" maxOccurs="unbounded" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>

                <xs:element name="Profilok">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Profil" type="ProfilTipus" minOccurs="0" maxOccurs="unbounded" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="Parkanyok">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Parkany" type="ParkanyTipus" minOccurs="0" maxOccurs="unbounded" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>

                <xs:element name="Anyagok">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Anyag" type="AnyagTipus" minOccurs="0" maxOccurs="unbounded" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>

            </xs:sequence>
        </xs:complexType>
```

```
        <xs:selector xpath="Rendelesek/Rendeles" />
        <xs:field xpath="@rendelesszam" />
    </xs:key>

    <xs:key name="profilID_fk">
        <xs:selector xpath="Profilok/Profil" />
        <xs:field xpath="@profilID" />
    </xs:key>

    <xs:key name="parkanyID_fk">
        <xs:selector xpath="Parkanyok/Parkany" />
        <xs:field xpath="@parkanyID" />
    </xs:key>

    <xs:keyref name="rendelesszam_fk-Uveg" refer="rendelesszam_fk">
        <xs:selector xpath="Uvegek/Uveg/rendelesszam_fk" />
        <xs:field xpath="." />
    </xs:keyref>

    <xs:keyref name="rendelesszam_fk-Beepites" refer="rendelesszam_fk">
        <xs:selector xpath="Beepitesek/Beepites/rendelesszam_fk" />
        <xs:field xpath="." />
    </xs:keyref>

    <xs:keyref name="rendelesszam_fk-Profil" refer="rendelesszam_fk">
        <xs:selector xpath="Profilok/Profil/rendelesszam_fk" />
        <xs:field xpath="." />
    </xs:keyref>

    <xs:keyref name="profilID_fk-Anyag" refer="profilID_fk">
        <xs:selector xpath="Anyagok/Anyag/profilID_fk" />
        <xs:field xpath="." />
    </xs:keyref>

    <xs:keyref name="parkanyID_fk-Anyag" refer="parkanyID_fk">
        <xs:selector xpath="Anyagok/Anyag/parkanyID_fk" />
        <xs:field xpath="." />
    </xs:keyref>

</xs:element>
```

```xml
<xs:complexType name="UvegTipus">
    <xs:sequence>
        <xs:element name="Retegszam" type="retegTipus" />
        <xs:element name="Hofolia" type="xs:string" />
        <xs:element name="rendelesszam_fk" type="xs:integer" />
    </xs:sequence>
    <xs:attribute name="uvegID" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:integer">
                <xs:pattern value="[0-9][0-9][0-9]" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>

<xs:complexType name="BeepitesTipus">
    <xs:sequence>
        <xs:element name="Datum" type="xs:string" />
        <xs:element name="Szakember" type="xs:string" />
        <xs:element name="rendelesszam_fk" type="xs:integer" />
    </xs:sequence>
    <xs:attribute name="beepID" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:integer">
                <xs:pattern value="[0-9][0-9][0-9]" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>
<xs:complexType name="RendelesTipus">
    <xs:sequence>
        <xs:element name="Rendeles_datum" type="xs:string" />
        <xs:element name="Megrendelo" type="megrendeloTipus" />
    </xs:sequence>
    <xs:attribute name="rendelesszam" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:integer">
                <xs:pattern value="[0-9][0-9][0-9]" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>
```

```
<xs:complexType name="ProfilTipus">
    <xs:sequence>
        <xs:element name="Meret" type="xs:int" />
        <xs:element name="Anyag" type="anyagTipus" />
        <xs:element name="rendelesszam_fk" type="xs:int"></xs:element>
    </xs:sequence>
    <xs:attribute name="profilID" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:integer">
                <xs:pattern value="[0-9][0-9][0-9]" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>

<xs:complexType name="ParkanyTipus">
    <xs:sequence>
        <xs:element name="Hossz" type="xs:int" />
        <xs:element name="Szelesseg" type="xs:int" />
    </xs:sequence>
    <xs:attribute name="parkanyID" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:integer">
                <xs:pattern value="[0-9][0-9][0-9]" />
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>

<xs:complexType name="AnyagTipus">
    <xs:sequence>
        <xs:element name="parkanyID_fk" type="xs:int" />
        <xs:element name="profilID_fk" type="xs:int" />
        <xs:element name="AzonosSzin" type="xs:string"></xs:element>
    </xs:sequence>
</xs:complexType>
```

```
    <xs:complexType name="retegTipus">
        <xs:sequence>
            <xs:element name="Haromreteg" type="xs:int" />
            <xs:element name="Otreteg" type="xs:int" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="megrendeloTipus">
        <xs:sequence>
            <xs:element name="KapcsoaltfelvetelDatum" type="xs:string" />
            <xs:element name="Nev" type="xs:string" />
            <xs:element name="Telepules" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="anyagTipus">
        <xs:sequence>
            <xs:element name="Muanyag" type="xs:int" />
            <xs:element name="Fa" type="xs:int" />
            <xs:element name="Aluminium" type="xs:int" />
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

3) Második feladat

3.1) Adatolvasás

Az adatolvasáshoz szükséges az XML fájl elérése, majd a parse művelet elvégzése. Ezt követően tudjuk az adatokat kezelni. A kiírás bizonyos lépései a formázás miatt kerültek szétbontásra.

```
static StringBuilder result = new StringBuilder();

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

    // Instantiate the Factory
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

    File toParse = new File("XMLM95ETT.xml");

    DocumentBuilder db = dbf.newDocumentBuilder();

    // parse XML file
    Document doc = db.parse(toParse);

    Node first = doc.getFirstChild();
    NodeList rest = first.getChildNodes();

    // write out the parent element's name
    result.append("Root Element: " + first.getNodeName());
    result.append("\n-------------------------");

    // call the method that reads out the file
    printNote(rest);

    // call the method that writes the result to a file
    writeTofile(result);

    System.out.println(result);

}
     private static void printNote(NodeList nodeList) {
         for (int count = 0; count < nodeList.getLength(); count++) {

             Node tempNode = nodeList.item(count);

             // make sure it's element node.
             if (tempNode.getNodeType() == Node.ELEMENT_NODE) {
                 // get node name and value
                 result.append("\n\nKategoria: " + tempNode.getNodeName());
                 result.append("\n-------------------------");

                 if (tempNode.hasAttributes()) {
                     // get attributes names and values
                     NamedNodeMap nodeMap = tempNode.getAttributes();
                     for (int i = 0; i < nodeMap.getLength(); i++) {
                         Node node = nodeMap.item(i);
                         result.append("\n\n" + node.getNodeName() + node.getNodeValue());
                     }
                 }
```

```
            //this means there are other elements in the node
            if (tempNode.getChildNodes().getLength() != 1) {
                NodeList childs = tempNode.getChildNodes();


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

                    Node asd = childs.item(i);

                    if (asd.getNodeType() == Node.ELEMENT_NODE) {
                        if (asd.getChildNodes().getLength() != 1) {

                            NodeList secondarychilds = asd.getChildNodes();

                            result.append("\n\n  " + asd.getNodeName());
                            result.append("\n-------------------------");

                            handleChildNodes(secondarychilds);

                        } else
                            result.append("\n  " + asd.getNodeName() + ": " + asd.getTextContent());
                    }
                }

            } else
                result.append("\n" + tempNode.getNodeName() + ": " + tempNode.getTextContent());
        }
    }
}
    //if there are multiple child nodes, this method handles them
    private static void handleChildNodes(NodeList childs) {
        for (int i = 0; i < childs.getLength(); i++) {

            Node nthChild = childs.item(i);

            if (nthChild.getNodeType() == Node.ELEMENT_NODE) {
                if (nthChild.getChildNodes().getLength() != 1) {
                    NodeList qwe = nthChild.getChildNodes();
                    result.append("\n    " + nthChild.getNodeName());
                    handleChildNodes(qwe);
                } else
                    result.append("\n  " + nthChild.getNodeName() + ": " + nthChild.getTextContent());
            }
        }
    }

    //create a FileWriter, and create/overwrite the file given the name
    private static void writeTofile(StringBuilder result) throws IOException {
        FileWriter myWriter = new FileWriter("DomReadM95ETT.txt");
        myWriter.write(result.toString());
        myWriter.close();
    }
}
```

3.2) Adatmódosítás

Az adatmódosító osztályban az elemek attribútumait, illetve az elemek tartalmát módosíthatjuk. Erre két külön osztályt hoztam létre. Az attribútumok egyediek, ezért azokat külön ellenőrizni nem szükséges. Az egyedek elérése xPath segítségével történik, így adategyezés esetén is biztosan a helyes elemet módosítjuk.

```java
public class DomModifyM95ETT {

    static Document readResult;

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

        //get the file for parsing
        File xmlName = new File("XMLM95ETT.xml");

        // call the method that parses the file
        readResult = readDoc(xmlName);

        //methods to modify the file - select ones you want to run
        //*********************************************************
        //modifyAttribute(readResult, "Profil", "profilID", "003", "321");
        //modifyAttribute(readResult, "Beepites", "beepID", "034", "340");
        //modifyElement(readResult, "/Ablakgyar/Uvegek/Uveg[@uvegID=147]/Hofolia", "3", "5");
        //modifyElement(readResult, "/Ablakgyar/Rendelesek/Rendeles[@rendelesszam=185]/Megrendelo/Telepules", "Miskolc", "Kazincbarcika");
        modifyElement(readResult, "/Ablakgyar/Anyagok/Anyag/profilID_fk", "012", "004");

        // write the result to a new file, and display it on console
        File myFile = new File("XMLM95ETTmod.xml");
        writeXml(readResult, myFile);
    }

    //parsing the input file
    public static Document readDoc(File xmlName) throws ParserConfigurationException, SAXException, IOException {

        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

        File toParse = xmlName;
        DocumentBuilder db = dbf.newDocumentBuilder();

        // parse XML file
        Document doc = db.parse(toParse);

        return (doc);
    }
    //modifies the selected attributes (as ID-s are unique, no type checking needed)
    public static void modifyAttribute(Document doc, String tagName, String attrName, String oldData, String newData) {

        NodeList nodes = doc.getElementsByTagName(tagName);

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

            Node node = nodes.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {

                if (node.getAttributes().getNamedItem(attrName).getTextContent().equals(oldData)) {

                    node.getAttributes().getNamedItem(attrName).setTextContent(newData);

                }
            }
        }
    }
    //modifies the selected element (using xPhath to avoid problems when two elements have the same attribute
    public static void modifyElement(Document doc, String accesPath, String oldData, String newData) throws XPathExpressionException {

        XPath xPath = XPathFactory.newInstance().newXPath();

        // get the nodelist based on the query
        NodeList nodes = (NodeList) xPath.compile(accesPath).evaluate(doc, XPathConstants.NODESET);

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

            Node node = nodes.item(i);

            if (node.getNodeType() == Node.ELEMENT_NODE) {

                if (node.getTextContent().equals(oldData)) {

                    node.setTextContent(newData);
                }
            }
        }
    }
```

```
    //writes the modified file to console, and to the specified file path
    // https://mkyong.com/java/how-to-modify-xml-file-in-java-dom-parser/
    private static void writeXml(Document doc, File output) throws TransformerException, UnsupportedEncodingException {

        TransformerFactory transformerFactory = TransformerFactory.newInstance();

        Transformer transformer = transformerFactory.newTransformer();

        // pretty print
        transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");

        DOMSource source = new DOMSource(doc);

        StreamResult console = new StreamResult(System.out);
        StreamResult file = new StreamResult(output);

        transformer.transform(source, console);
        transformer.transform(source, file);

    }
}
```

3.3) Adatlekérdezés

A lekérdezés során a módosító programhoz hasonlóan xPath változó értéke alapján keressük meg a megadott feltételeknek megfelelő elemeket.

```
public class DomQueryM95ETT {

    // class level static variables
    static StringBuilder readLines = new StringBuilder();
    static Document readResult;

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

        // get the file we need to work on
        File xmlName = new File("XMLM95ETT.xml");

        // call the method that parses the file
        readResult = readDoc(xmlName);

        // All queries - please select one, then run the program
        // ****************************************************
        // printNote(readResult, "/Ablakgyar/Uvegek/Uveg");
        // printNote(readResult, "/Ablakgyar/Uvegek/Uveg[@uvegID=146]");
        // printNote(readResult, "/Ablakgyar/Rendelesek/Rendeles[last()]");
        // printNote(readResult, "/Ablakgyar/Rendelesek/Rendeles[@rendelesszam > 183]");
        printNote(readResult, "/Ablakgyar/Profilok/Profil/Anyag[Fa > 1]");

        // write the query result to console
        System.out.println(readLines);

    }

    public static Document readDoc(File xmlName) throws ParserConfigurationException, SAXException, IOException {

        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

        File toParse = xmlName;
        DocumentBuilder db = dbf.newDocumentBuilder();

        // parse XML file
        Document doc = db.parse(toParse);

        // get the root element, then display it
        Node first = doc.getDocumentElement();

        readLines.append("Root Element: " + first.getNodeName());
        readLines.append("\n-------------------------");

        return (doc);
    }
```

```java
private static void printNote(Document doc, String queryName) throws XPathExpressionException {

    XPath xPath = XPathFactory.newInstance().newXPath();

    // get the nodelist based on the query
    NodeList elementsToRead = (NodeList) xPath.compile(queryName).evaluate(doc, XPathConstants.NODESET);

    // iterate trough the list, then write out the element nodes (if no child nodes
    // present, write out the text value as well
    for (int i = 0; i < elementsToRead.getLength(); i++) {

        Node tempNode = elementsToRead.item(i);

        // make sure it's element node.
        if (tempNode.getNodeType() == Node.ELEMENT_NODE) {

            readLines.append("\n\n  " + tempNode.getNodeName());
            readLines.append("\n  ----------------------");

            if (tempNode.hasAttributes()) {

                // get attributes names and values
                NamedNodeMap nodeMap = tempNode.getAttributes();
                for (int j = 0; j < nodeMap.getLength(); j++) {
                    Node node = nodeMap.item(j);
                    readLines.append("\n    " + node.getNodeName() + ": " + node.getNodeValue());
                }
            }

            if (tempNode.getChildNodes().getLength() != 1) {
                NodeList childs = tempNode.getChildNodes();

                for (int k = 0; k < childs.getLength(); k++) {

                    Node asd = childs.item(k);

                    if (asd.getNodeType() == Node.ELEMENT_NODE) {

                        handleChilds(asd);
                    }
                }

            } else
                readLines.append("\n" + tempNode.getNodeName() + ": " + tempNode.getTextContent());
        }
    }
}
```

```java
  private static void handleChilds(Node asd) {

      if (asd.getChildNodes().getLength() != 1) {
          NodeList secondarychilds = asd.getChildNodes();
          readLines.append("\n    " + asd.getNodeName());

          for (int l = 0; l < secondarychilds.getLength(); l++) {

              Node qwe = secondarychilds.item(l);

              if (qwe.getNodeType() == Node.ELEMENT_NODE) {

                  readLines.append("\n      " + qwe.getNodeName() + ": " + qwe.getTextContent());
              }
          }

      } else
          readLines.append("\n    " + asd.getNodeName() + ": " + asd.getTextContent());
  }
}
```