JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

MVK Zrt.

Készítette: **Kolozsvári Patrik**

Netünkön: **WYQ5JK**

Dátum: **2023.12.05.**

Tartalomjegyzék

A feladat leírása

A témámat a miskolci tömegközlekedés ihlette. A közlekedési társaság hivatalos neve: MVK Miskolc Városi Közlekedési Zártkörűen Működő Részvénytársaság, rövidebben MVK Zrt. A megbízható útitársunknak mint jogi személynek, van egy cégjegyzékszáma (05-10-000147), ami által nyilvántartható a cégjegyzékben a cégbíróság által. A tömegközlekedés megkívánja a megállóhelyek biztosítását város szerte, hogy az utasok könnyedén eljuthassanak az úti céljukhoz. A társaság autóbusz- és villamosközlekedést - valamint egy kisvasút vonalat is - biztosít, viszont az ER modellemben csak a buszok kaptak helyet, mivel ezek hangsúlyosabbak a közlekedés szempontjából, például az egyetemet is busszal lehet megközelíteni.

A sofőrök több buszt vezetnek, több útvonalon közlekednek. A munkabeosztás miatt lényeges dokumentálni, hogy melyik járművet melyik alkalmazott vezeti és mikor. Az MVK Zrt. a Miskolc Holding Önkormányzati Vagyonkezelő Zrt. tagvállalata (tulajdona), ami az alkalmazottakat különféle jutalmakban, bónuszukban részesíti, ehhez pedig szükséges a családi állapot ismerete. Például a sofőrök családtagjai számára éves kombinált bérletet biztosítanak.

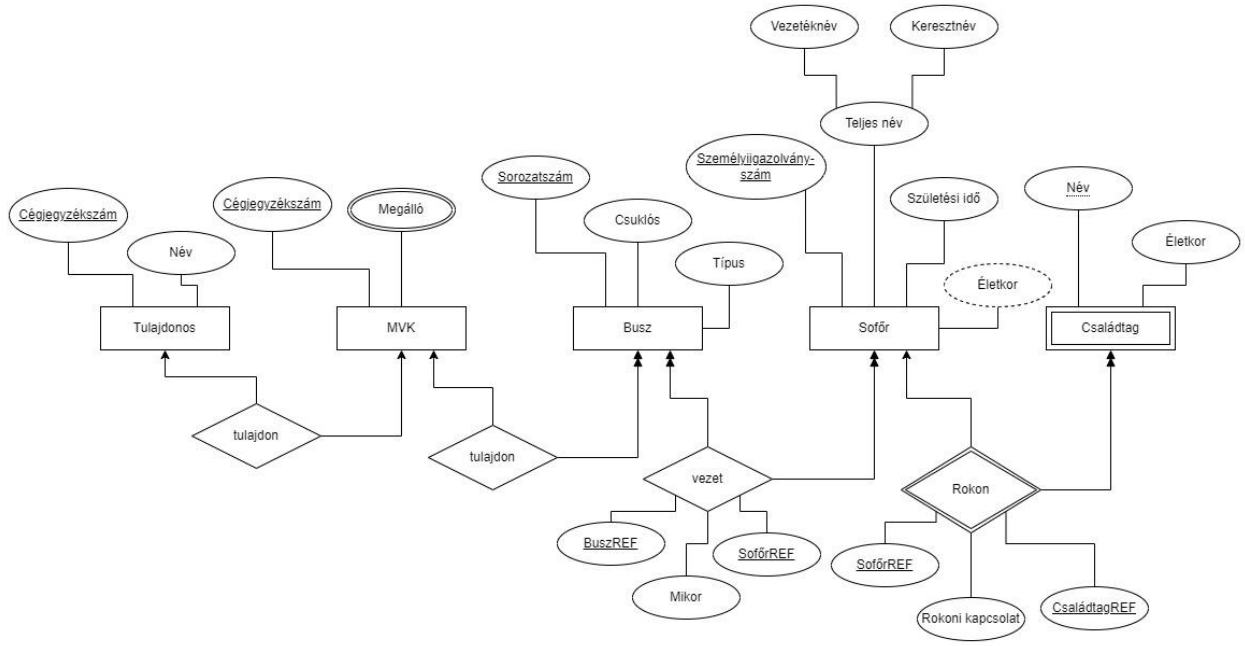
1. feladat

1a) Az adatbázis ER modell tervezése

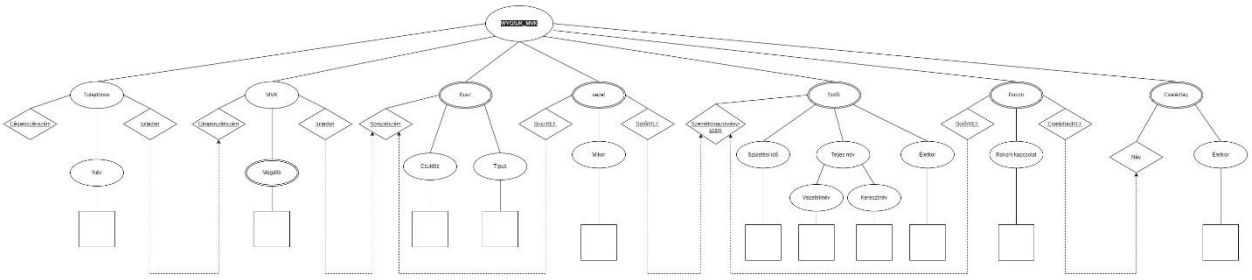
Az MVK Zrt.-nek több autóbusza van, így 1:N a köztük lévő reláció

Egy buszt több sofőr is vezet, valamint egy sofőr több buszt is vezet, így a köztük lévő reláció N:M

Egy alkalmazottnak (sofőr) több családtagja is lehet, viszont a családtagok csak egy sofőrhöz köthetőek, így a köztük lévő reláció 1:N



1b) Az adatbázis konvertálása XDM modellre



1c) Az XDM modell alapján XML dokumentum készítése

```
<?xml version="1.0" encoding="UTF-8"?>
<WYQ5JK_MVK xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
   xs:noNamespaceSchemaLocation="XMLSchemaWYQ5JK.xsd">
   <!-- TulAjdonos -->
   <tulajdonos t_cegjegyzekszam="05-10-000406">
      <name>Miskolc Holding Önkormányzati Vagyonkezelő Zrt.</name>
   </tulajdonos>
   <!-- MVK -->
   <mvk mvk_cegjegyzekszam="05-10-000147" t_cegjegyzekszam="05-10-000406">
      <megallok>
                  <megallo>Egyetemváros</megallo>
         <megallo>Centrum</megallo>
                  <megallo>Búza tér</megallo>
         </megallok>
   </mvk>
   <!-- Busz -->
   <busz sorozatszam="123">
      <csuklos>true</csuklos>
      <tipus>MAN Lion’s City GL A40 CNG</tipus>
   </busz>
   <busz sorozatszam="456">
      <csuklos>false</csuklos>
      <tipus>MAN Lion’s City A21 CNG</tipus>
   </busz>
   <busz sorozatszam="789">
      <csuklos>true</csuklos>
      <tipus>Neoplan N4522 Centroliner</tipus>
   </busz>
```

```
<!-- Sofőr -->
<sofor szemelyi="000001HE">
   <szuletesi_ido>1965-10-15</szuletesi_ido>
        <teljes_nev>
                <vezeteknev>Kovács</vezeteknev>
                <keresztnev>Béla</keresztnev>
        </teljes_nev>
        <eletkor>58</eletkor>
</sofor>
<sofor szemelyi="000002SA">
   <szuletesi_ido>1972-12-04</szuletesi_ido>
        <teljes_nev>
                <vezeteknev>Szabó</vezeteknev>
                <keresztnev>Tamás</keresztnev>
        </teljes_nev>
        <eletkor>51</eletkor>
</sofor>
<sofor szemelyi="000003TA">
   <szuletesi_ido>1971-04-29</szuletesi_ido>
        <teljes_nev>
                <vezeteknev>Tóth</vezeteknev>
                <keresztnev>István</keresztnev>
        </teljes_nev>
        <eletkor>52</eletkor>
</sofor>
<!-- Vezet -->
<vezet buszREF="123" soforREF="000002SA">
   <mikor>2023-11-13</mikor>
</vezet>
```

```
  <vezet buszREF="456" soforREF="000003TA">
    <mikor>2023-11-17</mikor>
  </vezet>
  <vezet buszREF="789" soforREF="000001HE">
    <mikor>2023-11-15</mikor>
  </vezet>
  <!-- Családtag -->
  <csaladtag nev="Kovácsné Mária">
    <kor>55</kor>
  </csaladtag>

  <csaladtag nev="Szabó Noémi">
    <kor>7</kor>
  </csaladtag>

  <csaladtag nev="Tóth Géza">
    <kor>12</kor>
  </csaladtag>
  <!-- Rokon -->
  <rokon soforREF="000001HE" csaladtagREF="Kovácsné Mária">
    <rokoni_kapcsolat>házastárs</rokoni_kapcsolat>
  </rokon>
  <rokon soforREF="000002SA" csaladtagREF="Szabó Noémi">
    <rokoni_kapcsolat>gyermek</rokoni_kapcsolat>
  </rokon>
  <rokon soforREF="000003TA" csaladtagREF="Tóth Géza">
    <rokoni_kapcsolat>gyermek</rokoni_kapcsolat>
  </rokon>
</WYQ5JK_MVK>
```

1d) Az XML dokumentum alapján XMLSchema készítése

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<!-- Elemek, tulajdonságok -->
	<xs:element name="name" type="xs:string" />
	<xs:element name="megallo" type="xs:string" />
	<xs:element name="csuklos" type="xs:boolean" />
	<xs:element name="tipus" type="xs:string" />
	<xs:element name="szuletesi_ido" type="datum_type" />
	<xs:element name="eletkor" type="xs:integer" />
	<xs:element name="mikor" type="datum_type" />
	<xs:element name="kor" type="xs:integer" />
	<xs:element name="rokoni_kapcsolat" type="xs:string" />
	<xs:attribute name="t_cegjegyzekszam" type="xs:string" />
	<xs:attribute name="mvk_cegjegyzekszam" type="xs:string" />
	<xs:attribute name="sorozatszam" type="xs:integer" />
	<xs:attribute name="szemelyi" type="xs:string" />
	<xs:attribute name="nev" type="xs:string" />
	<!-- Egyszerű típus -->
	<xs:simpleType name="datum_type">
		<xs:restriction base="xs:string">
			<xs:pattern value="(19|20)\d\d-(0[1-9]|1[012])-(0[1-9]|[12][0-
9]|3[01])"></xs:pattern>
		</xs:restriction>
	</xs:simpleType>
```

```
<!-- Komplex típusok -->
	<xs:complexType name="teljes_nev_type">
	<xs:sequence>
		<xs:element name="vezeteknev" type="xs:string" />
		<xs:element name="keresztnev" type="xs:string" />
	</xs:sequence>
	</xs:complexType>
	<xs:complexType name="megallok_type">
		<xs:sequence>
			<xs:element ref="megallo" minOccurs="3" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="tulajdonos_type">
		<xs:sequence>
			<xs:element ref="name" maxOccurs="1"/>
		</xs:sequence>
		<xs:attribute ref="t_cegjegyzekszam" use="required"/>
	</xs:complexType>
	<xs:complexType name="mvk_type">
		<xs:sequence>
			<xs:element name="megallok" type="megallok_type"/>
		</xs:sequence>
		<xs:attribute ref="mvk_cegjegyzekszam" use="required"/>
		<xs:attribute ref="t_cegjegyzekszam" use="required"/>
	</xs:complexType>
	<xs:complexType name="busz_type">
		<xs:sequence>
			<xs:element ref="csuklos"/>
			<xs:element ref="tipus"/>
```

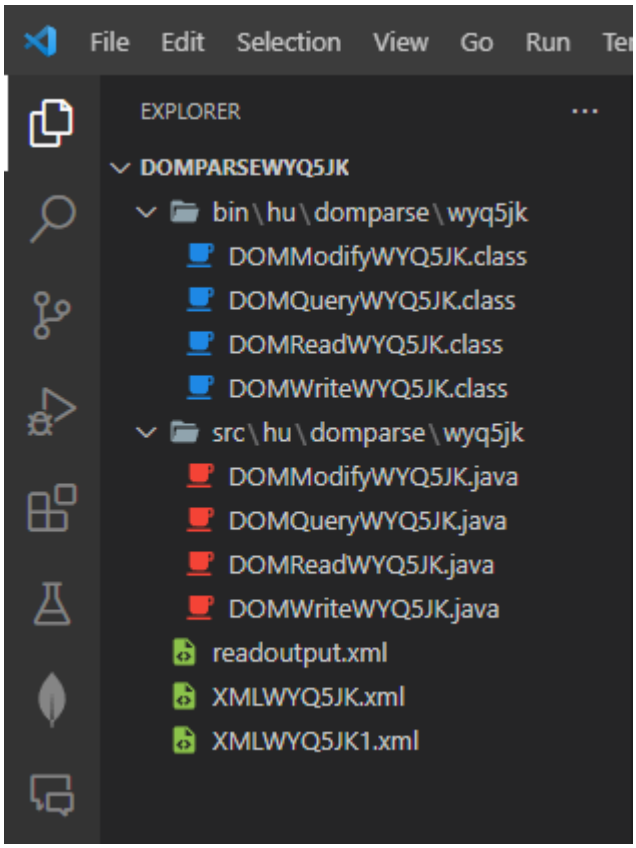
```
        </xs:sequence>
        <xs:attribute ref="sorozatszam" use="required"/>
</xs:complexType>
<xs:complexType name="sofor_type">
        <xs:sequence>
                <xs:element ref="szuletesi_ido"/>
                <xs:element name="teljes_nev" type="teljes_nev_type" />
                <xs:element ref="eletkor"/>
        </xs:sequence>
        <xs:attribute ref="szemelyi" use="required"/>
</xs:complexType>
<xs:complexType name="csaladtag_type">
        <xs:sequence>
                <xs:element ref="kor"/>
        </xs:sequence>
        <xs:attribute ref="nev" use="required"/>
</xs:complexType>
<!-- Kapcsolótáblák -->
<xs:complexType name="vezet_type">
        <xs:sequence>
        <xs:element ref="mikor"/>
</xs:sequence>
        <xs:attribute name="buszREF" type="xs:integer" use="required"/>
        <xs:attribute name="soforREF" type="xs:string" use="required"/>
</xs:complexType>
<xs:complexType name="rokon_type">
        <xs:sequence>
                <xs:element ref="rokoni_kapcsolat"/>
        </xs:sequence>
```

```
                <xs:attribute name="soforREF" type="xs:string" use="required"/>
                <xs:attribute name="csaladtagREF" type="xs:string" use="required"/>
        </xs:complexType>
        <xs:element name="WYQ5JK_MVK">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="tulajdonos" type="tulajdonos_type"
maxOccurs="1"/>
                                <xs:element name="mvk" type="mvk_type" maxOccurs="1"/>
                                <xs:element name="busz" type="busz_type" minOccurs="3"
maxOccurs="unbounded"/>
                                <xs:element name="sofor" type="sofor_type" minOccurs="3"
maxOccurs="unbounded"/>
                                <xs:element name="vezet" type="vezet_type"
maxOccurs="unbounded"/>
                                <xs:element name="csaladtag" type="csaladtag_type"
maxOccurs="unbounded"/>
                                <xs:element name="rokon" type="rokon_type"
maxOccurs="unbounded"/>

                        </xs:sequence>
                </xs:complexType>
          <!-- Elsődleges kulcsok -->
      <xs:unique name="t_cegjegyzekszam">
        <xs:selector xpath="tulajdonos"/>
        <xs:field xpath="@t_cegjegyzekszam"/>
      </xs:unique>
      <xs:unique name="mvk_cegjegyzekszam">
        <xs:selector xpath="mvk"/>
        <xs:field xpath="@mvk_cegjegyzekszam"/>
      </xs:unique>
```

```
    <xs:unique name="sorozatszam">
      <xs:selector xpath="busz"/>
      <xs:field xpath="@sorozatszam"/>
    </xs:unique>
    <xs:unique name="szemelyi">
      <xs:selector xpath="sofor"/>
      <xs:field xpath="@szemelyi"/>
    </xs:unique>
    <xs:unique name="nev">
      <xs:selector xpath="csaladtag"/>
      <xs:field xpath="@nev"/>
    </xs:unique>
        <!-- Idegen kulcsok -->
        <xs:keyref name="buszREF" refer="sorozatszam">
                <xs:selector xpath="busz"></xs:selector>
                <xs:field xpath="@buszREF"></xs:field>
        </xs:keyref>
        <xs:keyref name="soforREF" refer="szemelyi">
                <xs:selector xpath="sofor"></xs:selector>
                <xs:field xpath="@termekREF"></xs:field>
        </xs:keyref>
        <xs:keyref name="csaladtagREF" refer="nev">
                <xs:selector xpath="csaladtag"></xs:selector>
                <xs:field xpath="@termekREF"></xs:field>
        </xs:keyref>
        </xs:element>
</xs:schema>
```

2. feladat


File
Edit
Selection
View
Go
Run
EXPLORER
DOMPARSEWYQ5JK
bin \ hu \ domparse \ wyq5jk
DOMModifyWYQ5JK.class
DOMQueryWYQ5JK.class
DOMReadWYQ5JK.class
DOMWriteWYQ5JK.class
src \ hu \ domparse \ wyq5jk
DOMModifyWYQ5JK.java
DOMQueryWYQ5JK.java
DOMReadWYQ5JK.java
DOMWriteWYQ5JK.java
readoutput.xml
XMLWYQ5JK.xml
XMLWYQ5JK1.xml

2a) DOMRead

XML beolvasása, aztán output file létrehozása. Konzolra és a fájlba írás párhuzamosan. Először a gyökérelemet írattam ki, ezt követően függvények segítségével lekértem az elemeket és ezeket is kiírattam, majd lezártam a gyökérelemet. A kiíratásokat két függvénnyel oldottam meg, az egyik node listákat ír ki -- printNodeList() --, a másik – printNode() -- az egyes node-ok kiírásáért felelős. Plusz egy segédfüggvényt is használtam a behúzások miatt -- getIndentString()

DOMReadWYQ5JK.java

src > hu > domparse > wyq5jk > DOMReadWYQ5JK.java

```java
package hu.domparse.wyq5jk;

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

public class DOMReadWYQ5JK {
    Run | Debug
    public static void main(String[] args) {
        try {
            // Reading in the file
            File xmlFile = new File(pathname:"XMLWYQ5JK.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);
            doc.getDocumentElement().normalize();

            File outputFile = new File(pathname:"readoutput.xml");
            PrintWriter writer = new PrintWriter(new FileWriter(outputFile, append:true));

            // Printing out the root element and the prolog
            Element rootElement = doc.getDocumentElement();
            String rootName = rootElement.getTagName();
            StringJoiner rootAttributes = new StringJoiner(delimiter:" ");
            NamedNodeMap rootAttributeMap = rootElement.getAttributes();

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

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

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

            // Processing the elements
            NodeList tulajdonosList = doc.getElementsByTagName(tagname:"tulajdonos");
            NodeList mvkList = doc.getElementsByTagName(tagname:"mvk");
            NodeList buszList = doc.getElementsByTagName(tagname:"busz");
            NodeList soforList = doc.getElementsByTagName(tagname:"sofor");
            NodeList vezetList = doc.getElementsByTagName(tagname:"vezet");
            NodeList csaladtagList = doc.getElementsByTagName(tagname:"csaladtag");
            NodeList rokonList = doc.getElementsByTagName(tagname:"rokon");

```

```
47              // Printing out the (formatted) XML to the console and to the file
48              printNodeList(tulajdonosList, writer);
49              printNodeList(mvkList, writer);
50              printNodeList(buszList, writer);
51              printNodeList(soforList, writer);
52              printNodeList(vezetList, writer);
53              printNodeList(csaladtagList, writer);
54              printNodeList(rokonList, writer);
55
56              // Closing the root element
57              System.out.println("</" + rootName + ">");
58              writer.append("</" + rootName + ">");
59
60              writer.close();
61          } catch (Exception e) {
62              e.printStackTrace();
63          }
64      }
65
66      // Printing out the content of the NodeList recursively
67      private static void printNodeList(NodeList nodeList, PrintWriter writer) {
68          for (int i = 0; i < nodeList.getLength(); i++) {
69              Node node = nodeList.item(i);
70              printNode(node, indent:1, writer);
71              System.out.println(x:"");
72              writer.println(x:"");
73          }
74          System.out.println(x:"");
75          writer.println(x:"");
76      }
77
```

```
78      // Printing out the content of the Node recursively
79      private static void printNode(Node node, int indent, PrintWriter writer) {
80          if (node.getNodeType() == Node.ELEMENT_NODE) {
81              Element element = (Element) node;
82              String nodeName = element.getTagName();
83              StringJoiner attributes = new StringJoiner(delimiter:" ");
84              NamedNodeMap attributeMap = element.getAttributes();
85
86              // Name and attributes
87              for (int i = 0; i < attributeMap.getLength(); i++) {
88                  Node attribute = attributeMap.item(i);
89                  attributes.add(attribute.getNodeName() + "=\"" + attribute.getNodeValue() + "\"");
90              }
91
92              System.out.print(getIndentString(indent));
93              System.out.print("<" + nodeName + " " + attributes.toString() + ">");
94
95              writer.print(getIndentString(indent));
96              writer.print("<" + nodeName + " " + attributes.toString() + ">");
97
98              // Content
99              NodeList children = element.getChildNodes();
100             if (children.getLength() == 1 && children.item(index:0).getNodeType() == Node.TEXT_NODE) {
101                 System.out.print(children.item(index:0).getNodeValue());
102                 writer.print(children.item(index:0).getNodeValue());
103             } else {
104                 System.out.println();
105                 writer.println();
106                 for (int i = 0; i < children.getLength(); i++) {
107                     printNode(children.item(i), indent + 1, writer);
108                 }
109                 System.out.print(getIndentString(indent));
110                 writer.print(getIndentString(indent));
111             }
112
113             // Closing the node
114             System.out.println("</" + nodeName + ">");
115             writer.println("</" + nodeName + ">");
116         }
117     }
118
```

```
119     // Indenting
120     private static String getIndentString(int indent) {
121         StringBuilder sb = new StringBuilder();
122         for (int i = 0; i < indent; i++) {
123             sb.append(str:"  "); // 2 spaces per indent level
124         }
125         return sb.toString();
126     }
127 }
```

2b) DOMModify

Módosítások:

- Tulajdonos nevének megváltoztatása
- Egy busz típusának megváltoztatása
- Egy sofőr életkorának megváltoztatása
- Az egyik vezetés dátumának megváltoztatása

A módosított XML stringgé alakítása és konzolra történő kiíratása

DOMModifyWYQ5JK.java

src > hu > domparse > wyq5jk > DOMModifyWYQ5JK.java > {} hu.domparse.wyq5jk

```java
package hu.domparse.wyq5jk;

import org.w3c.dom.*;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
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.StringWriter;

public class DOMModifyWYQ5JK {
    Run | Debug
    public static void main(String[] args) {
        try {
            // Reading in the file
            File xmlFile = new File(pathname:"XMLWYQ5JK.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);


            // Implementation of changes

            //Changing the name of the owner
            NodeList tulajdonosList = doc.getElementsByTagName(tagname:"tulajdonos");
            Element tulajdonos = (Element) tulajdonosList.item(index:0);
            tulajdonos.getElementsByTagName(name:"name").item(index:0).setTextContent(textContent:"Kolozsvári Patrik");

            //Changing the type of the bus
            NodeList buszList = doc.getElementsByTagName(tagname:"busz");
            Element busz = (Element) buszList.item(index:1);
            busz.getElementsByTagName(name:"tipus").item(index:0).setTextContent(textContent:"nem rossz busz");

            //Changing the age of a driver
            NodeList soforList = doc.getElementsByTagName(tagname:"sofor");
            Element sofor = (Element) soforList.item(index:1);
            sofor.getElementsByTagName(name:"eletkor").item(index:0).setTextContent(textContent:"30");

            //Changing the date of an element in Vezet
            NodeList vezetList = doc.getElementsByTagName(tagname:"vezet");
            Element vezet = (Element) vezetList.item(index:1);
            vezet.getElementsByTagName(name:"mikor").item(index:0).setTextContent(textContent:"2023-11-14");
```

```
44
45
46              // Transforming the modified document to a string
47              TransformerFactory transformerFactory = TransformerFactory.newInstance();
48              Transformer transformer = transformerFactory.newTransformer();
49
50              // Creating a StringWriter to write the XML string
51              StringWriter writer = new StringWriter();
52              transformer.transform(new DOMSource(doc), new StreamResult(writer));
53
54              // Getting the modified XML as a string
55              String output = writer.toString();
56              // Printing out the modified XML to the console
57              System.out.println(output);
58
59          } catch (Exception e) {
60              e.printStackTrace();
61          }
62      }
63  }
64
```

2c) DOMQuery

Lekérdezések:

- Az összes sofőr neve
- A „789”-es sorozatszámú busz adatai
- A legalább 52 éves sofőrök nevei
- A Szabó Tamás által vezetett busz típusa

DOMQueryWYQ5JK.java

src > hu > domparse > wyq5jk > DOMQueryWYQ5JK.java > ...

```java
package hu.domparse.wyq5jk;

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

public class DOMQueryWYQ5JK {
    Run | Debug
    public static void main(String[] args) {
        try {
            // Reading in the file
            File xmlFile = new File(pathname:"XMLWYQ5JK.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);

            // Queries
            firstQuery(doc);
            secondQuery(doc);
            thirdQuery(doc);
            fourthQuery(doc);

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

    private static void firstQuery(Document doc)
    {
        // Querying the name of all drivers
        NodeList soforList = doc.getElementsByTagName(tagname:"sofor");
        System.out.println(x:"1, Soforok: ");
        for (int i = 0; i < soforList.getLength(); i++) {
            Element sofor = (Element) soforList.item(i);
            String vezeteknev = sofor.getElementsByTagName(name:"vezeteknev").item(index:0).getTextContent();
            String keresztnev = sofor.getElementsByTagName(name:"keresztnev").item(index:0).getTextContent();
            System.out.println(vezeteknev + " " + keresztnev);
        }
        System.out.println();
    }

```

```java
    private static void secondQuery(Document doc)
    {
        // Querying the data of the bus with serial number '789'
        String sorozatszam = "789";
        NodeList buszList = doc.getElementsByTagName(tagname:"borrower");
        for (int i = 0; i < buszList.getLength(); i++) {
            Element busz = (Element) buszList.item(i);
            if (busz.getAttribute(name:"sorozatszam").equals(sorozatszam)) {
                String csuklos = busz.getElementsByTagName(name:"csuklos").item(index:0).getTextContent();
                String tipus = busz.getElementsByTagName(name:"tipus").item(index:0).getTextContent();

                System.out.println("2, Data of the bus with serial number '" + sorozatszam + "':");
                System.out.println("Csuklos: " + csuklos);
                System.out.println("Tipus: " + tipus);
            }
        }
        System.out.println();
    }
```

```java
    private static void thirdQuery(Document doc)
    {
        // Querying the name of the driver(s) who are at least 52 years old
        NodeList soforList = doc.getElementsByTagName(tagname:"sofor");
        System.out.println(x:"3, Drivers who are at least 52 years old:");
        for (int i = 0; i < soforList.getLength(); i++) {
            Element sofor = (Element) soforList.item(i);
            int age = Integer.parseInt(sofor.getElementsByTagName(name:"age").item(index:0).getTextContent());

            if (age >= 52) {
                String vezeteknev = sofor.getElementsByTagName(name:"vezeteknev").item(index:0).getTextContent();
                String keresztnev = sofor.getElementsByTagName(name:"keresztnev").item(index:0).getTextContent();
                System.out.println(vezeteknev + " " + keresztnev);
            }
        }
        System.out.println();
    }

    private static void fourthQuery(Document doc)
    {
        // Querying the type of the bus dtiven by Szabó Tamás
        String sofornev = "Szabó Tamás";

        NodeList soforList = doc.getElementsByTagName(tagname:"sofor");
        for (int i = 0; i < soforList.getLength(); i++) {
            Element sofor = (Element) soforList.item(i);
            String vezeteknev = sofor.getElementsByTagName(name:"vezeteknev").item(index:0).getTextContent();
            String keresztnev = sofor.getElementsByTagName(name:"keresztnev").item(index:0).getTextContent();
            String soforteljesnev = vezeteknev + " " + keresztnev;

            if (soforteljesnev.equals(sofornev)) {
                String szemelyi = sofor.getAttribute(name:"szemelyi");

                NodeList vezetList = doc.getElementsByTagName(tagname:"vezet");
                System.out.println("5, Type of the bus driven by " + sofornev + ": ");
                for (int j = 0; j < vezetList.getLength(); j++) {
                    Element vezet = (Element) vezetList.item(j);
                    if (vezet.getAttribute(name:"szemelyi").equals(szemelyi)) {
                        String sorozatszam = vezet.getAttribute(name:"sorozatszam");

                        NodeList buszList = doc.getElementsByTagName(tagname:"busz");
                        for (int k = 0; k < buszList.getLength(); k++) {
                            Element book = (Element) buszList.item(k);
                            if (book.getAttribute(name:"sorozatszam").equals(sorozatszam)) {
                                String tipus = book.getElementsByTagName(name:"tipus").item(index:0).getTextContent();
                                System.out.println(tipus);
                            }
                        }
```

2d) DOMWrite

Új dokumentum létrehozása, adatokkal való feltöltése.

addData(): létrehozza a gyökérelemet, majd meghívja az egyes elemeket létrehozó függvényeket, a különböző adatokkal paraméterként

Ezután az 2a) feladatban alkalmazott módon, strukturált formában konzolra íratás és egy fájlba az XML dokumentumot

DOMWriteWYQ5JK.java

src > hu > domparse > wyq5jk > DOMWriteWYQ5JK.java > ...

```java
package hu.domparse.wyq5jk;

import java.io.File;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.util.StringJoiner;

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

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

public class DOMWriteWYQ5JK {
    Run | Debug
    public static void main(String[] args) {
        try {
            // Creating a new document
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.newDocument();

            // Filling up the document with data
            addData(doc);

            File outputFile = new File(pathname:"XMLWYQ5JK1.xml");
            PrintWriter writer = new PrintWriter(new FileWriter(outputFile, append:true));

            // Printing out the root element and the prolog
            Element rootElement = doc.getDocumentElement();
            String rootName = rootElement.getTagName();
            StringJoiner rootAttributes = new StringJoiner(delimiter:" ");
            NamedNodeMap rootAttributeMap = rootElement.getAttributes();

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

```
41
42          System.out.println(x:"<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
43          writer.println(x:"<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
44
45          System.out.println("<" + rootName + " " + rootAttributes.toString() + ">\n");
46          writer.println("<" + rootName + " " + rootAttributes.toString() + ">\n");
47
48          // Processing the elements
49          NodeList tulajdonosList = doc.getElementsByTagName(tagname:"tulajdonos");
50          NodeList mvkList = doc.getElementsByTagName(tagname:"mvk");
51          NodeList buszList = doc.getElementsByTagName(tagname:"busz");
52          NodeList soforList = doc.getElementsByTagName(tagname:"sofor");
53          NodeList vezetList = doc.getElementsByTagName(tagname:"vezet");
54          NodeList csaladtagList = doc.getElementsByTagName(tagname:"csaladtag");
55          NodeList rokonList = doc.getElementsByTagName(tagname:"rokon");
56
57          // Printing out the (formatted) XML to the console and to the file
58          printNodeList(tulajdonosList, writer);
59          printNodeList(mvkList, writer);
60          printNodeList(buszList, writer);
61          printNodeList(soforList, writer);
62          printNodeList(vezetList, writer);
63          printNodeList(csaladtagList, writer);
64          printNodeList(rokonList, writer);
65
66          // Closing the root element
67          System.out.println("</" + rootName + ">");
68          writer.append("</" + rootName + ">");
69
70          writer.close();
71      } catch (Exception e) {
72          e.printStackTrace();
73      }
74  }
```

```java
    private static void addData(Document doc) {

        Element rootElement = doc.createElement(tagName:"WYQ5JK_MVK");
        rootElement.setAttribute(name:"xmlns:xs", value:"http://www.w3.org/2001/XMLSchema-instance");
        rootElement.setAttribute(name:"xs:noNamespaceSchemaLocation", value:"XMLSchemaWYQ5JK.xsd");
        doc.appendChild(rootElement);

        addTulajdonos(doc, rootElement, tcegjegyzekszam:"05-10-000406", name:"Miskolc Holding Önkormányzati Vagyonkezelő Zrt.");

        addMvk(doc, rootElement, mvkcegjegyzekszam:"05-10-000147", tcegjegyzekszam:"05-10-000406", megallo1:"Egyetemváros", megallo2:"Centrum", megallo3:"Búza tér");

        addBusz(doc, rootElement, sorozatszam:"123", csuklos:"true", tipus:"MAN Lion?s City GL A40 CNG");
        addBusz(doc, rootElement, sorozatszam:"456", csuklos:"false", tipus:"MAN Lion?s City A21 CNG");
        addBusz(doc, rootElement, sorozatszam:"789", csuklos:"true", tipus:"Neoplan N4522 Centroliner");

        addSofor(doc, rootElement, szemelyi:"000001HE", szuletesiido:"1965-10-15", vezeteknev:"Kovács", keresztnev:"Béla", eletkor:"58");
        addSofor(doc, rootElement, szemelyi:"000002SA", szuletesiido:"1972-12-04", vezeteknev:"Szabó", keresztnev:"Tamás", eletkor:"51");
        addSofor(doc, rootElement, szemelyi:"000003TA", szuletesiido:"1971-04-29", vezeteknev:"Tóth", keresztnev:"István", eletkor:"52");

        addVezet(doc, rootElement, buszref:"123", soforref:"000002SA", mikor:"2023-11-13");
        addVezet(doc, rootElement, buszref:"456", soforref:"000003TA", mikor:"2023-11-17");
        addVezet(doc, rootElement, buszref:"789", soforref:"000001HE", mikor:"2023-11-15");

        addCsaladtag(doc, rootElement, nev:"Kovácsné Mária", kor:"55");
        addCsaladtag(doc, rootElement, nev:"Szabó Noémi", kor:"7");
        addCsaladtag(doc, rootElement, nev:"Tóth Géza", kor:"12");

        addRokon(doc, rootElement, soforref:"000001HE", csaladtagref:"Kovácsné Mária", rokonikapcsolat:"házastárs");
        addRokon(doc, rootElement, soforref:"000002SA", csaladtagref:"Szabó Noémi", rokonikapcsolat:"gyermek");
        addRokon(doc, rootElement, soforref:"000003TA", csaladtagref:"Tóth Géza", rokonikapcsolat:"gyermek");

    }
    private static void addTulajdonos(Document doc, Element rootElement, String tcegjegyzekszam, String name) {
        Element tulajdonos = doc.createElement(tagName:"tulajdonos");
        tulajdonos.setAttribute(name:"t_cegjegyzekszam", tcegjegyzekszam);

        Element nameElement = createElement(doc, name:"name", name);
        tulajdonos.appendChild(nameElement);

        rootElement.appendChild(tulajdonos);
    }

    private static void addMvk(Document doc, Element rootElement, String mvkcegjegyzekszam, String tcegjegyzekszam, String megallo1, String megallo2, String megallo3) {
        Element mvk = doc.createElement(tagName:"mvk");
        mvk.setAttribute(name:"mvk_cegjegyzekszam", mvkcegjegyzekszam);
        mvk.setAttribute(name:"t_cegjegyzekszam", tcegjegyzekszam);

        Element megallok = doc.createElement(tagName:"megallok");
        Element megallo1Element = createElement(doc, name:"megallo", megallo1);
        megallok.appendChild(megallo1Element);
        Element megallo2Element = createElement(doc, name:"megallo", megallo2);
        megallok.appendChild(megallo2Element);
        Element megallo3Element = createElement(doc, name:"megallo", megallo3);
        megallok.appendChild(megallo3Element);

        rootElement.appendChild(mvk);
    }

    private static void addBusz(Document doc, Element rootElement, String sorozatszam, String csuklos, String tipus) {
        Element busz = doc.createElement(tagName:"busz");
        busz.setAttribute(name:"sorozatszam", sorozatszam);

        Element csuklosElement = createElement(doc, name:"csuklos", csuklos);
        busz.appendChild(csuklosElement);

        Element tipusElement = createElement(doc, name:"tipus", tipus);
        busz.appendChild(tipusElement);

        rootElement.appendChild(busz);
    }
```

```java
    private static void addSofor(Document doc, Element rootElement, String szemelyi, String szuletesiido, String vezeteknev, String keresztnev, String eletkor) {
        Element sofor = doc.createElement(tagName:"sofor");
        sofor.setAttribute(name:"szemelyi", szemelyi);

        Element szuletesiidoElement = createElement(doc, name:"szuletesi_ido", szuletesiido);
        sofor.appendChild(szuletesiidoElement);

        Element name = doc.createElement(tagName:"teljes_nev");
        Element vezeteknevElement = createElement(doc, name:"vezeteknev", vezeteknev);
        Element keresztnevElement = createElement(doc, name:"keresztnev", keresztnev);
        name.appendChild(vezeteknevElement);
        name.appendChild(keresztnevElement);

        Element eletkorElement = createElement(doc, name:"eletkor", eletkor);
        sofor.appendChild(eletkorElement);

        rootElement.appendChild(sofor);
    }

    private static void addVezet(Document doc, Element rootElement, String buszref, String soforref, String mikor) {
        Element vezet = doc.createElement(tagName:"vezet");
        vezet.setAttribute(name:"buszREF", buszref);
        vezet.setAttribute(name:"soforREF", soforref);

        Element mikorElement = createElement(doc, name:"mikor", mikor);
        vezet.appendChild(mikorElement);

        rootElement.appendChild(vezet);
    }

    private static void addCsaladtag(Document doc, Element rootElement, String nev, String kor) {
        Element csaladtag = doc.createElement(tagName:"csaladtag");
        csaladtag.setAttribute(name:"nev", nev);

        Element korElement = createElement(doc, name:"kor", kor);
        csaladtag.appendChild(korElement);

        rootElement.appendChild(csaladtag);
    }
```

```java
    private static void addRokon(Document doc, Element rootElement, String soforref, String csaladtagref, String rokonikapcsolat) {
        Element rokon = doc.createElement(tagName:"rokon");
        rokon.setAttribute(name:"soforREF", soforref);
        rokon.setAttribute(name:"csaladtagREF", csaladtagref);

        Element rokonikapcsolaElement = createElement(doc, name:"rokoni_kapcsolat", rokonikapcsolat);
        rokon.appendChild(rokonikapcsolaElement);

        rootElement.appendChild(rokon);
    }


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

    // Printing out the content of the NodeList recursively
    private static void printNodeList(NodeList nodeList, PrintWriter writer) {
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            printNode(node, indent:1, writer);
            System.out.println(x:"");
            writer.println(x:"");
        }
        System.out.println(x:"");
        writer.println(x:"");
    }

```

```
218     // Printing out the content of the Node recursively
219     private static void printNode(Node node, int indent, PrintWriter writer) {
220         if (node.getNodeType() == Node.ELEMENT_NODE) {
221           Element element = (Element) node;
222           String nodeName = element.getTagName();
223           StringJoiner attributes = new StringJoiner(delimiter:" ");
224           NamedNodeMap attributeMap = element.getAttributes();
225 
226           // Name and attributes
227           for (int i = 0; i < attributeMap.getLength(); i++) {
228             Node attribute = attributeMap.item(i);
229             attributes.add(attribute.getNodeName() + "=\"" + attribute.getNodeValue() + "\"");
230           }
231 
232           System.out.print(getIndentString(indent));
233           System.out.print("<" + nodeName + " " + attributes.toString()+ ">");
234 
235           writer.print(getIndentString(indent));
236           writer.print("<" + nodeName + " " + attributes.toString() + ">");
237 
238           // Content
239           NodeList children = element.getChildNodes();
240           if (children.getLength() == 1 && children.item(index:0).getNodeType() == Node.TEXT_NODE) {
241             System.out.print(children.item(index:0).getNodeValue());
242             writer.print(children.item(index:0).getNodeValue());
243           } else {
244             System.out.println();
245             writer.println();
246             for (int i = 0; i < children.getLength(); i++) {
247               printNode(children.item(i), indent + 1, writer);
248             }
249             System.out.print(getIndentString(indent));
250             writer.print(getIndentString(indent));
251           }
252 
253           // Closing the node
254           System.out.println("</" + nodeName + ">");
255           writer.println("</" + nodeName + ">");
256         }
257     }
258 
```

```
259     // Indenting
260     private static String getIndentString(int indent) {
261         StringBuilder sb = new StringBuilder();
262         for (int i = 0; i < indent; i++) {
263           sb.append(str:"  "); // 2 spaces per indent level
264         }
265         return sb.toString();
266     }
267 }
268 
```