

JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Gépjármű-felépítés

Készítette: **Bettes Benjámín László**

Neptunkód: **IG1TYM**

Dátum: **2022. 11. 13.**

Tartalomjegyzék

1. Feladat	3
1a) Az adatbázis ER modell	3
1b) Az adatbázis konvertálása XDM modellre	4
1c) Az XDM modell alapján XML dokumentum készítése	4
1d) Az XML dokumentum alapján XMLSchema készítése	9
2. Feladat – DOM program készítés	14
2a) Adatolvasás, kiírás konzolra – fájlba	14
2b) Adatmódosítás	23
2c) Adatlekérdezés	31
DOM	31
XPath	34

A feladat leírása: Egy gépjármű felépítésének XML környezetben való megvalósítása. Egy személyautó főbb komponensei az egyedek, gyerekelemei az ezekhez kapcsolódó tulajdonságok, a kapcsolatok pedig az alkotórészek közötti összefüggések. Gyökér elemként a `gepjarmu-felepites` elem szolgál. Ennek gyerekelemei a többi elemek (`auto`, `gyarto`, `modell`, `motor`, `gyartja`). A „`gyartja`” elem a N:M kapcsolat tulajdonsága alapján lett létrehozva, egy „`evjarat`” gyerekelemmel. A többértékű tulajdonságok legalább 3 elemmel rendelkeznek a dokumentumban. A származtatott teljesítmény attribútum rendes gyerekelemként jelenik meg, külön értékkel.

1. Feladat

1a) Az adatbázis ER modell

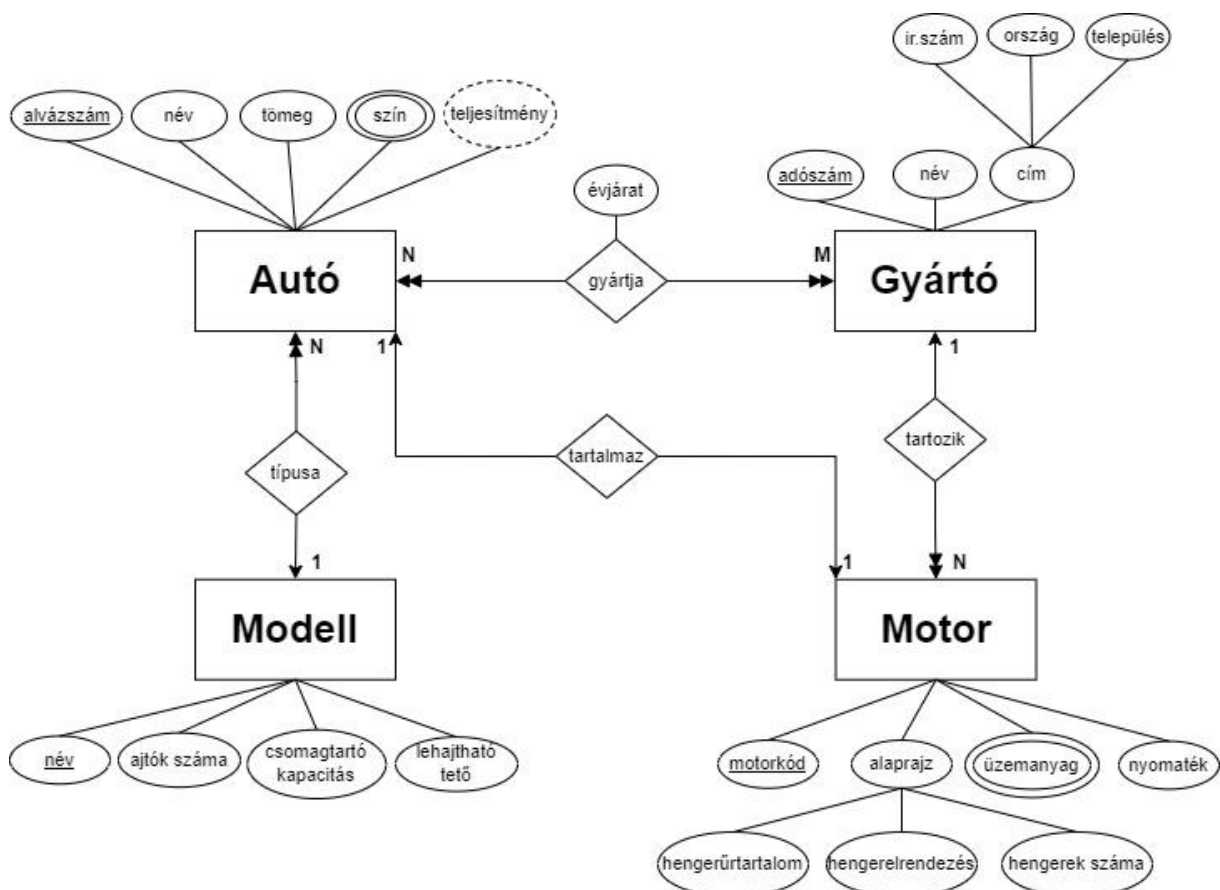
A **teljesítmény attribútum** származtatott, mivel a motor nyomatékából, illetve az autó tömegéből számolható.

Gyártja kapcsolat: Egy cég több autót is gyárthat, viszont egy autón több cég is dolgozhat (az autótörténelemben több példa is megfigyelhető).

Tartozik kapcsolat: Egy gyártóhoz több motor is tartozhat, de egy motort csak egy gyártó szabadalmaztat.

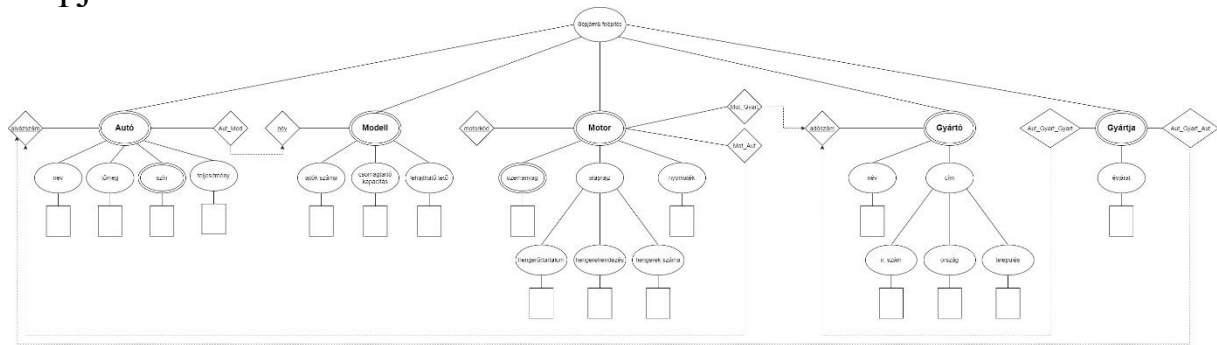
Tartalmaz kapcsolat: Egy autóban egyszerre csak egy motor lehet, illetve egy motor, mint egy egyed csak egy autóban található meg.

Típusa kapcsolat: Egy modell típusból több autót is gyártanak, ugyanakkor egy járművet pedig egyszerre csak egy modell jellemezhet.



1b) Az adatbázis konvertálása XDM modellre

A modellt az órán bemutattak, és a gyakorlaton elkészített mintapélda alapján készítettem el.



1c) Az XDM modell alapján XML dokumentum készítése

A konverziós szabályokat követve ismét a gyakorlaton megoldott feladatok alapján készítettem az XML dokumentumot. Minden többértékű gyerekelemből legalább 3 darab létezik. Idegen kulcsok nevének szabványa pl.: Aut_Mod - Az autó idegen kulcsa ami a modellel köti össze (a modell valamelyik egyedére mutat a kulcs értéke által). Avagy az autóból a modellre mutató kulcs.

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

```
<gepjarmu-felepites xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="XSDIG1TYM.xsd">
```

```
<!-- Autók -->
```

 $\langle \! | \text{-----} \rangle$

```
<auto alvazszam="01" Aut_Mod="3-Door Hatchback">
```

<nev>Sprinter Trueno GT-Apex</nev>

<tomeg>970</tomeg>

<szin>Panda</szin>

<szin>Yellow</szin>

<szin>Wine Metallic</szin>

<szin>White Metallic</szin>

<teljesitmeny>123 HP</teljesitmeny>

</auto>

<auto alvazszam="02" Aut_Mod="5-Door Hatchback">

<nev>206 Profil</nev>

<tomeg>1100</tomeg>

<szin>Indigo Blue</szin>

<szin>Rogue Red</szin>

<szin>Vert Green</szin>

<teljesitmeny>60 HP</teljesitmeny>

</auto>

<auto alvazszam="03" Aut_Mod="Coupe">

<nev>Skyline GTR R34</nev>

<tomeg>1560</tomeg>

<szin>Bayside Blue</szin>

<szin>Midnight Purple</szin>

<szin>Millenium Jade</szin>

<szin>Lightning Yellow</szin>

<szin>Active Red</szin>

<szin>Pearl White</szin>

<teljesitmeny>276 HP</teljesitmeny>

</auto>

<!-- ----->

<!-- Gyártók -->

<!-- ----->

<gyarto adoszam="123">

<nev>Toyota</nev>

<cim>

<orszag>Japán</orszag>

<iranyitoszam>1234</iranyitoszam>

<telepules>Toiota</telepules>

```
</cim>
</gyarto>

<gyarto adoszam="124">
  <nev>Peugeot</nev>
  <cim>
    <ország>Franciaország</ország>
    <iranyitoszam>1235</iranyitoszam>
    <telepules>Párizs</telepules>
  </cim>
</gyarto>

<gyarto adoszam="125">
  <nev>Nissan</nev>
  <cim>
    <ország>Japán</ország>
    <iranyitoszam>1326</iranyitoszam>
    <telepules>Yokohama</telepules>
  </cim>
</gyarto>
<!-- ----->

<!-- Modellek -->
<!-- ----->
<modell nev="3-Door Hatchback">
  <ajtok_szama>3</ajtok_szama>
  <csomag tarto_meret>280</csomag tarto_meret>
  <lehajthat o_teto>Nincs</lehajthat o_teto>
</modell>

<modell nev="5-Door Hatchback">
  <ajtok_szama>5</ajtok_szama>
```

```
<csomag tarto_meret>237</csomag tarto_meret>
<lehajthato_teto>Nincs</lehajthato_teto>
</modell>
```

```
<modell nev="Coupe">
  <ajtok_szama>2</ajtok_szama>
  <csomag tarto_meret>315</csomag tarto_meret>
  <lehajthato_teto>Nincs</lehajthato_teto>
</modell>
```

```
<!-- ----->
```

```
<!-- Motorok -->
```

```
<!-- ----->
```

```
<motor motorkod="4A-GE" Mot_Aut="01" Mot_Gyart="123">
  <alaprajz>
    <hengerurtartalom>1598</hengerurtartalom>
    <hengerelrendezes>I</hengerelrendezes>
    <hengerek_szama>4</hengerek_szama>
  </alaprajz>
  <uzemanyag>95 Benzin E10</uzemanyag>
  <uzemanyag>98 Benzin E5</uzemanyag>
  <uzemanyag>100 Benzin E5</uzemanyag>
  <nyomatek>92 kW</nyomatek>
</motor>
```

```
<motor motorkod="TU1JP" Mot_Aut="02" Mot_Gyart="124">
  <alaprajz>
    <hengerurtartalom>1110</hengerurtartalom>
    <hengerelrendezes>I</hengerelrendezes>
    <hengerek_szama>4</hengerek_szama>
  </alaprajz>
  <uzemanyag>95 Benzin E10</uzemanyag>
```

```
<uzemanyag>98 Benzin E5</uzemanyag>
<uzemanyag>100 Benzin E5</uzemanyag>
<nyomatek>44 kW</nyomatek>
</motor>

<motor motorkod="RB26DETT" Mot_Aut="03" Mot_Gyart="125">
  <alaprajz>
    <hengerurtartalom>2600</hengerurtartalom>
    <hengerelrendezes>I</hengerelrendezes>
    <hengerek_szama>6</hengerek_szama>
  </alaprajz>
  <uzemanyag>95 Benzin E10</uzemanyag>
  <uzemanyag>98 Benzin E5</uzemanyag>
  <uzemanyag>100 Benzin E5</uzemanyag>
  <nyomatek>206 kW</nyomatek>
</motor>

<!-- ----->
<!-- Gyártja kapcsoló -->
<!-- ----->
<gyartja Aut_Gyart_Aut="01" Aut_Gyart_Gyart="123">
  <evjarat>1986</evjarat>
</gyartja>

<gyartja Aut_Gyart_Aut="02" Aut_Gyart_Gyart="124">
  <evjarat>2000</evjarat>
</gyartja>

<gyartja Aut_Gyart_Aut="03" Aut_Gyart_Gyart="125">
  <evjarat>2002</evjarat>
</gyartja>
<!-- ----->
</gepjarmu-felepites>
```


1d) Az XML dokumentum alapján XMLSchema készítése

A gyakorlaton megoldott étterem példa alapján készült. A tetoTipus simpleType korlátozva van a „Van” és „Nincs” szóra. Többször előforduló elem esetében a maxOccurs „unbounded”-ra van állítva.

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

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

```
  <xs:simpleType name="tetoTipus">
```

```
    <xs:restriction base="xs:token">
```

```
      <xs:enumeration value="van" />
```

```
      <xs:enumeration value="nincs" />
```

```
    </xs:restriction>
```

```
  </xs:simpleType>
```

```
  <!-- Egyszerű típusok - Felépítés -->
```

```
  <xs:element name="gepjarmu-felepites">
```

```
    <xs:complexType>
```

```
      <xs:sequence>
```

```
        <xs:element name="auto" maxOccurs="unbounded" />
```

```
        <xs:element name="gyarto" maxOccurs="unbounded" />
```

```
        <xs:element name="modell" maxOccurs="unbounded" />
```

```
        <xs:element name="motor" maxOccurs="unbounded" />
```

```
        <xs:element name="gyartja" maxOccurs="unbounded" />
```

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

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

```
  <!-- Kulcsok -->
```

```
  <xs:key name="auto_kulcs">
```

```
<xs:selector xpath="auto" />
<xs:field xpath="@alvazsam" />
</xs:key>
```

```
<xs:key name="gyarto_kulcs">
  <xs:selector xpath="gyarto" />
  <xs:field xpath="@adoszam" />
</xs:key>
```

```
<xs:key name="modell_kulcs">
  <xs:selector xpath="modell" />
  <xs:field xpath="@nev" />
</xs:key>
```

```
<xs:key name="motor_kulcs">
  <xs:selector xpath="motor" />
  <xs:field xpath="@motorkod" />
</xs:key>
```

```
<!-- Idegen kulcsok -->
```

```
<xs:keyref refer="auto_kulcs" name="auto_idegen_kulcs">
  <xs:selector xpath="gyartja" />
  <xs:field xpath="@Aut_Gyart_Aut" />
</xs:keyref>
```

```
<xs:keyref refer="gyarto_kulcs" name="gyarto_idegen_kulcs">
  <xs:selector xpath="gyartja" />
  <xs:field xpath="@Aut_Gyart_Gyart" />
</xs:keyref>
```

```
<xs:keyref refer="modell_kulcs" name="modell_auto_idegen_kulcs">
  <xs:selector xpath="auto" />
  <xs:field xpath="@Aut_Mod" />
</xs:keyref>
```

```
<xs:keyref refer="auto_kulcs" name="auto_motor_idegen_kulcs">
  <xs:selector xpath="motor" />
  <xs:field xpath="@Mot_Aut" />
</xs:keyref>
```

```
<xs:keyref refer="gyarto_kulcs" name="gyarto_motor_idegen_kulcs">
  <xs:selector xpath="motor" />
  <xs:field xpath="@Mot_Gyart" />
</xs:keyref>
```

```
</xs:element>
```

```
<xs:element name="auto">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nev" type="xs:string" />
      <xs:element name="tomeg" type="xs:integer" />
      <xs:element name="szin" type="xs:string" maxOccurs="unbounded" />
      <xs:element name="teljesitmeny" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="gyarto">
```

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="nev" type="xs:string" />
    <xs:element name="cim">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="ország" type="xs:string" />
          <xs:element name="iranyitoszam" type="xs:integer" />
          <xs:element name="telepules" type="xs:string" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

```
<xs:element name="modell">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ajtok_szama" type="xs:integer" />
      <xs:element name="csomag tarto_meret" type="xs:integer" />
      <xs:element name="lehajthato_teto" type="tetoTipus" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="motor">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="alaprajz">
```

```

    <xs:complexType>
      <xs:sequence>
        <xs:element name="hengerurtartalom" type="xs:integer" />
        <xs:element name="hengerelrendezes" type="xs:integer" />
        <xs:element name="hengerek_szama" type="xs:integer" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="uzemanyag" type="xs:string"
maxOccurs="unbounded" />
  <xs:element name="nyomatek" type="xs:string" />
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="gyartja">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="evjarat" type="xs:integer" />
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>

```

2. Feladat – DOM program készítés

2a) Adatolvasás, kiírás konzolra – fájlba

A 7. gyakorlaton vettek alapján készült. Ahol eltérés van a gyakorlati feladattól megjegyzéseket használtam. Ilyen például a szín elem, mivel többértékű, ezért egy for ciklussal olvasom ki mindegyiket, amely olyan hosszú, ahány olyan nevű elem létezik. Emellett, mivel a címnek több gyerekeleme is van, ezért csak fejlécként írom ki azt az elemet, a gyerekeit pedig egyesével alatta.

```
package hu.domparse.IG1TYM;

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

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

import org.xml.sax.SAXException;

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

public class DOMReadIG1TYM
{
    public static void main(String argv[]) throws SAXException, IOException,
    ParserConfigurationException
    {
```

```
// XML illetve output TXT fájl létrehozása
File xmlFile = new File("XMLIG1TYM.xml");
File outputFile = new File("outputFile.txt");

// Fájlíró létrehozása az output TXT-hez
FileWriter fWriter = new FileWriter(outputFile, false);

javax.xml.parsers.DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = factory.newDocumentBuilder();

// DOM fa előállítás
Document doc = dBuilder.parse(xmlFile);

doc.getDocumentElement().normalize();
// Normalizálás segít a helyes elemek elérésében

// Gyökérelem kiírása
String root = doc.getDocumentElement().getNodeName();
System.out.println("Gyökér elem: " + root);
fWriter.write("Gyökér elem: " + root + "\n");

// Adott nevű elem kilistázása
NodeList nList = doc.getElementsByTagName("auto");

for(int i = 0; i < nList.getLength(); i++)
{
    Node nNode = nList.item(i);

    // Jelenlegi elem nevének kiolvasása, majd megjelenítése konzolon, kiírása a
    fájlba
```

```

String currentNode = nNode.getNodeName();
System.out.println("\nJelenlegi elem: " + currentNode);
fWriter.write("\nJelenlegi elem: " + currentNode);

if (nNode.getNodeType() == Node.ELEMENT_NODE)
{
    Element elem = (Element) nNode;

    // Alvázszámm attribútum kiolvasása
    String alvaz = elem.getAttribute("alvazszam");

    // Név és tömeg elemek kiolvasása majd kiírása
    Node node1 = elem.getElementsByTagName("nev").item(0);
    String nev = node1.getTextContent();

    Node node2 = elem.getElementsByTagName("tomeg").item(0);
    String tomeg = node2.getTextContent();

    System.out.println("Alvázszám: " + alvaz);
    System.out.println("Név: " + nev);
    System.out.println("Tömeg: " + tomeg);
    fWriter.write("\nAlvázszám: " + alvaz + "\nNév: " + nev + "\nTömeg: " +
tomeg + "\n");

    // Mivel a szín elem többértékű, ezért egy for ciklussal olvasom ki
mindegyiket,
    // amely olyan hosszú, ahány olyan nevű elem létezik
    System.out.println("Elérhető színek: ");
    fWriter.write("Elérhető színek: ");
    for(int j = 0; j < elem.getElementsByTagName("szin").getLength(); j++)
    {

```



```

Node node3 = elem.getElementsByTagName("szin").item(j);
String szin = node3.getTextContent();
System.out.println(" " + szin);
fWriter.write("\n" + szin);
}

// Teljesítmény kiolvasása és kiírása
Node node4 = elem.getElementsByTagName("teljesitmeny").item(0);
String perf = node4.getTextContent();

System.out.println("Teljesítmény: " + perf);
fWriter.write("\nTeljesítmény: " + perf + "\n");
}
}

// Nem hozok létre új változót, hanem csak a listamutatót változtatom
nList = doc.getElementsByTagName("gyarto");

for(int i = 0; i < nList.getLength(); i++)
{
    Node nNode = nList.item(i);

    String currentNode = nNode.getNodeName();
    System.out.println("\nJelenlegi elem: " + currentNode);
    fWriter.write("\nJelenlegi elem: " + currentNode);

    if (nNode.getNodeType() == Node.ELEMENT_NODE)
    {
        Element elem = (Element) nNode;

```

```

String adoszam = elem.getAttribute("adoszam");

Node nodel = elem.getElementsByTagName("nev").item(0);
String nev = nodel.getTextContent();

System.out.println("Adószám: " + adoszam);
System.out.println("Név: " + nev);
fWriter.write("\nAdószám: " + adoszam + "\nNév: " + nev);

// Mivel a címnek több gyerekeleme is van, ezért csak fejlécként írom ki ezt
az elemet,
// a gyerekeit pedig egyesével alatta mint a többit
System.out.println("Cím: ");
fWriter.write("\nCím: ");
Node node2 = elem.getElementsByTagName("ország").item(0);
String orszag = node2.getTextContent();
System.out.println("Ország: " + orszag);

Node node3 = elem.getElementsByTagName("iranyitoszam").item(0);
String irszam = node3.getTextContent();
System.out.println("Írányítószám: " + irszam);

Node node4 = elem.getElementsByTagName("telepules").item(0);
String telepules = node4.getTextContent();
System.out.println("Település: " + telepules);

fWriter.write("\nOrszág: " + orszag + "\nÍrányítószám: " + irszam +
"\nTelepülés: " + telepules + "\n");
}
}

```

// Innentől kezdve a többi beolvasás is a fentebb említett módszereket követik

```
nList = doc.getElementsByTagName("modell");
```

```
for(int i = 0; i < nList.getLength(); i++)
```

```
{
```

```
    Node nNode = nList.item(i);
```

```
    String currentNode = nNode.getNodeName();
```

```
    System.out.println("\nJelenlegi elem: " + currentNode);
```

```
    fWriter.write("\nJelenlegi elem: " + currentNode);
```

```
    if (nNode.getNodeType() == Node.ELEMENT_NODE)
```

```
    {
```

```
        Element elem = (Element) nNode;
```

```
        String nev = elem.getAttribute("nev");
```

```
        System.out.println("Név: " + nev);
```

```
        fWriter.write("\nNév: " + nev);
```

```
        Node node1 = elem.getElementsByTagName("ajtok_szama").item(0);
```

```
        String ajtoksz = node1.getTextContent();
```

```
        System.out.println("Ajtók száma: " + ajtoksz);
```

```
        Node node2 =
```

```
elem.getElementsByTagName("csomag tarto _meret").item(0);
```

```
        String csomag tartom = node2.getTextContent();
```

```
        System.out.println("Csomag tartó mérete: " + csomag tartom);
```

```
        Node node3 = elem.getElementsByTagName("lehajthato _teto").item(0);
```

```
        String teto = node3.getTextContent();
```

```

        System.out.println("Lehajtható tető: " + teto);

        fWriter.write("\nAjtók száma: " + ajtoksz + "\nCsomagtartó mérete: " +
csomagtartom + "\nLehajtható tető:" + teto + "\n");
    }
}

nList = doc.getElementsByTagName("motor");

for(int i = 0; i < nList.getLength(); i++)
{
    Node nNode = nList.item(i);

    String currentNode = nNode.getNodeName();
    System.out.println("\nJelenlegi elem: " + currentNode);
    fWriter.write("\nJelenlegi elem: " + currentNode);

    if (nNode.getNodeType() == Node.ELEMENT_NODE)
    {
        Element elem = (Element) nNode;

        String motorkod = elem.getAttribute("motorkod");
        System.out.println("Morokód: " + motorkod);
        fWriter.write("\nMotorkód: " + motorkod);

        System.out.println("Alaprajz: ");
        fWriter.write("\nAlaprajz: ");
        Node node1 = elem.getElementsByTagName("hengerurtartalom").item(0);
        String urtartalom = node1.getTextContent();
        System.out.println("Hengerűrtartalom: " + urtartalom);
    }
}

```

```

Node node2 = elem.getElementsByTagName("hengerelrendezes").item(0);
String elrendezes = node2.getTextContent();
System.out.println("Hengerelrendezés: " + elrendezes);

Node node3 = elem.getElementsByTagName("hengerek_szama").item(0);
String hengerszam = node3.getTextContent();
System.out.println("Hengerek száma: " + hengerszam);

fWriter.write("\nHengerűrtartalom: " + urtartalom + "\nHengerelrendezés: "
+ elrendezes + "\nHengerek száma:" + hengerszam + "\n");

System.out.println("Üzemanyagok: ");
fWriter.write("Üzemanyagok: ");
for(int j = 0; j < elem.getElementsByTagName("uzemanyag").getLength();
j++)
{
Node node4 = elem.getElementsByTagName("uzemanyag").item(j);
String uzemanyag = node4.getTextContent();
System.out.println(" " + uzemanyag);
fWriter.write("\n" + uzemanyag);
}
fWriter.write("\n");
}
}

nList = doc.getElementsByTagName("gyartja");

for(int i = 0; i < nList.getLength(); i++)
{
Node nNode = nList.item(i);

```

```

String currentNode = nNode.getNodeName();
System.out.println("\nJelenlegi elem: " + currentNode);
fWriter.write("\nJelenlegi elem: " + currentNode);

if (nNode.getNodeType() == Node.ELEMENT_NODE)
{
    Element elem = (Element) nNode;

    String AGYA = elem.getAttribute("Aut_Gyart_Aut");
    String AGYGY = elem.getAttribute("Aut_Gyart_Gyart");

    Node node = elem.getElementsByTagName("evjarat").item(0);
    String evjarat = node.getTextContent();

    System.out.println("Autó alvázszám: " + AGYA);
    System.out.println("Gyártó adószám: " + AGYGY);
    System.out.println("Évjárat: " + evjarat);
    fWriter.write("\nAutó alvázszám: " + AGYA + "\nGyártó adószám: " +
AGYGY + "\nÉvjárat: " + evjarat + "\n");
}
}

// Fájlíró bezárása
fWriter.close();
}
}

```

2b) Adatmódosítás

Ismét a 7. gyakorlaton tanultak alapján készült. A kódban a különböző módosítások a

```
// ---  
// ***  
// ---
```

kommentek közé vannak bezárva, mindenhol a módosítás egy rövid leírásával kezdve. A végrehajtott módosításokat a végén kiírom a konzolra. Ilyen módosítás például a mezőérték változtatás, gyerekelem törlés, valamint egy új gyerekelem felvétele, kezdeti értékkel.

```
package hu.domparse.IG1TYM;
```

```
import java.io.File;
```

```
// Egyed (Transformer) osztály, factory (TransformerFactory) ami létrehozza ezt az  
egyedet és kivételek, amiket dobhatnak importálása
```

```
// Mivel a transzformációnak mindig van egy forrása és eredménye, kell azaz osztály,  
ami szükséges ahhoz,
```

```
// hogy a DOM-ot használjuk forrásként (DOMSource) és egy kimeneti folyam az  
eredményeknek (StreamResult)
```

```
import javax.xml.parsers.DocumentBuilderFactory;
```

```
import javax.xml.parsers.DocumentBuilder;
```

```
import javax.xml.transform.Transformer;
```

```
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.NodeList;
```

```
import org.w3c.dom.Node;
```

```
import org.w3c.dom.Element;
```

```

import org.w3c.dom.NamedNodeMap;

public class DOMModifyIG1TYM
{
    public static void main(String argv[])
    {
        try
        {
            File inputFile = new File("XMLIG1TYM.xml");

            DocumentBuilderFactory docFactory =
DocumentBuilderFactory.newInstance();

            DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

            Document doc = docBuilder.parse(inputFile);

            // Ezek csak a kód szétválasztása miatt vannak, minden rész az adott
            kódfejezet rövid leírásával kezd
            // -----
            -----

            //
            *****
            *****

            // -----
            -----

            // A "supercar" példához nagyon hasonlóan attribútum értéket, illetve mező
            értéket változtat,

            // azzal a kivétellel, hogy mivel nekem az "auto" egyedből több is van,

            // bele van téve még egy for loop-ba az egész, ami addig megy, ahány ilyen
            elemem van

            NodeList autoList = doc.getElementsByTagName("auto");

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

```



```

// Mindig változtatom az aktuálisan lekért auto egyedem
Node auto = doc.getElementsByTagName("auto").item(i);

// auto attribútumának módosítása
NamedNodeMap attr = auto.getAttributes();
// Ha az attribútum az alvázszám
Node nodeAttr = attr.getNamedItem("alvazszam");
// Akkor változtassa meg "A0X"-re, ahol X az auto egyed jelenlegi indexe a
sorban (+1 az indexelés miatt, mert az 0-tól indul,
// de én azt akarom hogy 1-től)
nodeAttr.setTextContent("A0" + (i+1));

// auto gyerekelemeinek kilistázása
NodeList list = auto.getChildNodes();

// for loop ami a gyerekelemek számáig megy
for(int temp = 0; temp < list.getLength(); temp++)
{
    Node node = list.item(temp);

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

        // A név nevű gyerekelemnél teljesül
        if ("nev".equals(eElement.getNodeName()))
        {
            // Ha a név egyenlő ezzel
            if ("Sprinter Trueno GT-Apex".equals(eElement.getTextContent()))
            {
                // Változtassa meg erre

```

```

        eElement.setTextContent("Sprinter Trueno GT-Apex AE86");
    }
    if ("206 Profil".equals(eElement.getTextContent()))
    {
        eElement.setTextContent("206 Profil 1.1");
    }
}
}
}

// -----
-----

//
*****
*****

// -----
-----

// A luxurycars törléséhez hasonló, ugyanazzal a különbséggel, mint az előbbi
módosításnál

// Itt a "modell" elemek "ajtok_szama" gyerekelemét törölöm
NodeList modellList = doc.getElementsByTagName("modell");
for(int i = 0; i < modellList.getLength(); i++)
{
    // Kilistázza a jelenlegi modell egyedet
    Node modell = doc.getElementsByTagName("modell").item(i);

    // Lekéri annak gyerekelemeit
    NodeList childNodes = modell.getChildNodes();

    // Végigmegy a gyerekelemeken
    for (int count = 0; count < childNodes.getLength(); count++)
    {

```

```

        Node node = childNodes.item(count);
        // Ha a gyerekelem neve "ajtok_szama"
        if("ajtok_szama".equals(node.getNodeName()))
        {
            // Akkor törölje
            modell.removeChild(node);
        }
    }
}

// -----
-----

//
*****
*****

// -----
-----

// Alaprajzon belül módosítja a hengerek rendezésbe írtakat
NodeList alaprajzList = doc.getElementsByTagName("alaprajz");
for(int i = 0; i < alaprajzList.getLength(); i++)
{
    // Kilistázza az alaprajz egyedeket
    Node alaprajz = doc.getElementsByTagName("alaprajz").item(i);

    // Lekéri annak gyerekelemeit
    NodeList childNodes = alaprajz.getChildNodes();

    // for loop ami a gyerekelemek számáig megy
    for(int temp = 0; temp < childNodes.getLength(); temp++)
    {
        Node node = childNodes.item(temp);

        // Ellenőrzés hogy a kapott egyed elem-e

```

```

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

    // A hengerelrendezés nevű gyerekelemnél teljesül
    if("hengerelrendezes".equals(eElement.getNodeName()))
    {
        // Ha a hengerelrendezés egyenlő ezzel
        if ("I".equals(eElement.getTextContent()))
        {
            // Változtassa meg erre
            eElement.setTextContent("Inline");
        }
    }
}

// -----
//
// -----
// -----
// -----

// Új gyerekelemet vesz fel a gyártja egyedbe - "honap", majd az
Aut_Gyart_Aut attribútum értéke alapján állít neki értéket
NodeList gyartjaList = doc.getElementsByTagName("gyartja");
for (int i = 0; i < gyartjaList.getLength(); i++)
{
    Node gyartja = gyartjaList.item(i);

    // Lekéri az "Aut_Gyart_Aut" attribútum értékét és eltárolja az "id"-ben

```

```
String id =  
gyartja.getAttributes().getNamedItem("Aut_Gyart_Aut").getTextContent();
```

```
// Létrehozza az új "honap" elemet
```

```
Element honap = doc.createElement("honap");
```

```
gyartja.appendChild(honap);
```

```
// Az "id" értéke alapján ad értéket az új "honap" elemnek
```

```
if ("01".equals(id))
```

```
{
```

```
    honap.appendChild(doc.createTextNode("03"));
```

```
}
```

```
if ("02".equals(id))
```

```
{
```

```
    honap.appendChild(doc.createTextNode("01"));
```

```
}
```

```
if ("03".equals(id))
```

```
{
```

```
    honap.appendChild(doc.createTextNode("08"));
```

```
}
```

```
}
```

```
// -----
```

```
//
```

```
*****  
*****
```

```
// -----
```

```
// Tratalom konzolra írása:
```

```
// Csinálunk egy transzformációt ami ahhoz kell, hogy az XML fájlt a  
System.out-ra továbbítsuk
```

```
// Létrehozunk egy transformer objektumot, használjuk a DOM-ot hogy  
létrehozzunk egy forrás objektumot és
```

```
// használjuk a System.out-ot hogy építsünk egy eredmény objektumot
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(doc);

System.out.println("-----Módosított fájl-----");
StreamResult consoleResult = new StreamResult(System.out);
transformer.transform(source, consoleResult);

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

2c) Adatlekérdezés

A DOM program mellé készült egy XPath verzió is a feladathoz, a 8. gyakorlat alapján. Ugyanis az XPath-ot logikusabbnak láttam a lekérdezések megvalósításának szempontjából. A gyakorlati feladathoz hasonlóan az XPath programban a lekérdezések ki vannak kommentezve, a megfelelő sor kikommentezésével lehet lefuttatni a programot.

DOM

```
package hu.domparse.IG1TYM;
```

```
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 DOMQueryIG1TYM
```

```
{
```

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

```
    {
```

```
        //Forrás file
```

```
        File file = new File("XMLIG1TYM.xml");
```

```
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
```

```
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
```

```
Document doc = dBuilder.parse(file);
```

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

```
// Gyökér elem
```

```
System.out.print("Gyökér elem: ");
```

```
System.out.println(doc.getDocumentElement().getNodeName());
```

```
// Autók kilistázása
```

```
NodeList nList = doc.getElementsByTagName("auto");
```

```
System.out.println("-----");
```

```
// Végigfut az "auto"-nak a gyerekelemein, kihagyva a "szin"-t
```

```
for(int i = 0; i < nList.getLength(); i++)
```

```
{
```

```
    Node node = nList.item(i);
```

```
    System.out.println("\nJelenlegi elem: " + node.getNodeName());
```

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

```
    {
```

```
        Element elem = (Element) node;
```

```
        System.out.println("Alvázsám: " + elem.getAttribute("alvazsam"));
```

```
        NodeList nList2 = elem.getChildNodes();
```

```
        for(int j = 0; j < nList2.getLength(); j++)
```

```
        {
```



```
Node node2 = nList2.item(j);

if(node2.getNodeType() == Node.ELEMENT_NODE)
{
    if(!node2.getNodeName().equals("szin"))
    {
        System.out.println(node2.getNodeName() + " : " +
node2.getTextContent());
    }
}
}
}
}
}
```

XPath

```
package hu.dompase.IG1TYM;

import java.io.IOException;

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

import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

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

import org.xml.sax.SAXException;

public class XPathQueryIG1TYM
{
    public static void main(String[] args)
    {
        try
        {
            // DocumentBuilder létrehozása

            DocumentBuilderFactory documentBuilderFactory =
                DocumentBuilderFactory.newInstance();
```

```
DocumentBuilder documentBuilder =  
documentBuilderFactory.newDocumentBuilder();
```

```
Document document = documentBuilder.parse("XMLIG1TYM.xml");
```

```
document.getDocumentElement().normalize();
```

```
// az XPath készítése
```

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

```
// -----  
-----
```

```
// LEKÉRDEZÉSEK
```

```
// -----  
-----
```

```
// Kiválasztja az autó utolsó elemét
```

```
//String expression = "/gepjarmu-felepites/auto[last()]";
```

```
// Kiválasztja azon autókat, amelyeknek a tömege legfeljebb 1200
```

```
//String expression = "//auto[tomeg<1200]";
```

```
// Kiválasztja a gyártó illetve modell elemeket
```

```
//String expression = "//gyarto | //modell";
```

```
// Kiválasztja az első és harmadik motort
```

```
//String expression = "//motor[1] | //motor[3]";
```

```
// Kiválasztja azt a gyártót, akinek adószáma 123
```

```
String expression = "//gyarto[@adoszam=123]";
```

```
// -----  
-----
```

// Készítsünk egy listát, majd a xPath kifejezést le kell fordítani és ki kell értékelni

```
NodeList nodeList = (NodeList)
XPath.compile(expression).evaluate(document, XPathConstants.NODESET);
```

// A for ciklus segítségével a NodeList csomópontajin végig kell iterálni

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

```
{
```

```
    Node node = nodeList.item(i);
```

```
    System.out.println("\nAktuális elem: " + node.getNodeName());
```

// Meg kell vizsgálni a csomópontot, tesztelni kell a subelemet

// autó csomópont

```
    if (node.getNodeType() == Node.ELEMENT_NODE &&
        node.getNodeName().equals("auto"))
```

```
    {
```

```
        Element element = (Element) node;
```

```
        System.out.println("Alvászám: " + element.getAttribute("alvazsam"));
```

```
        System.out.println("Név: " +
            element.getElementsByTagName("nev").item(0).getTextContent());
```

```
        System.out.println("Tömeg: " +
            element.getElementsByTagName("tomeg").item(0).getTextContent());
```

```
        for(int j = 0; j < element.getElementsByTagName("szin").getLength();
j++)
```

```
        {
```

```
            System.out.println("Szín: " +
                element.getElementsByTagName("szin").item(j).getTextContent());
```

```
        }
```

```

        System.out.println("Teljesítmény: " +
element.getElementsByTagName("teljesitmeny").item(0).getTextContent());
    }

    // gyártó csomópont
    if (node.getNodeType() == Node.ELEMENT_NODE &&
node.getNodeName().equals("gyarto"))
    {
        Element element = (Element) node;

        System.out.println("Adószám: " + element.getAttribute("adoszam"));

        System.out.println("Név: " +
element.getElementsByTagName("nev").item(0).getTextContent());

        System.out.println("Cím:");

        System.out.println("Ország: " +
element.getElementsByTagName("orszag").item(0).getTextContent());

        System.out.println("Írányítószám: " +
element.getElementsByTagName("iranyitoszam").item(0).getTextContent());

        System.out.println("Település: " +
element.getElementsByTagName("telepules").item(0).getTextContent());
    }

    // modell csomópont
    if (node.getNodeType() == Node.ELEMENT_NODE &&
node.getNodeName().equals("modell"))
    {
        Element element = (Element) node;

        System.out.println("Név: " + element.getAttribute("nev"));

        System.out.println("Ajtók száma: " +
element.getElementsByTagName("ajtok_szama").item(0).getTextContent());

        System.out.println("Csomagtartó méret: " +
element.getElementsByTagName("csomag tarto _meret").item(0).getTextContent());
    }

```

```

        System.out.println("Lehajtható tető: " +
element.getElementsByTagName("lehajthato_teto").item(0).getTextContent());
    }

    // motor csomópont
    if (node.getNodeType() == Node.ELEMENT_NODE &&
node.getNodeName().equals("motor"))
    {
        Element element = (Element) node;

        System.out.println("Motorkód: " + element.getAttribute("motorkod"));
        System.out.println("Alaprajz:");

        System.out.println("Hengerűrtartalom: " +
element.getElementsByTagName("hengerurtartalom").item(0).getTextContent());

        System.out.println("Hengerelrendezés: " +
element.getElementsByTagName("hengerelrendezes").item(0).getTextContent());

        System.out.println("Hengerek száma: " +
element.getElementsByTagName("hengerek_szama").item(0).getTextContent());

        for(int j = 0; j <
element.getElementsByTagName("uzemanyag").getLength(); j++)
        {
            System.out.println("Üzemanyag: " +
element.getElementsByTagName("uzemanyag").item(j).getTextContent());
        }

        System.out.println("Nyomaték: " +
element.getElementsByTagName("nyomatek").item(0).getTextContent());
    }

}

}

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

```

```
    catch (SAXException e) {e.printStackTrace();}
    catch (IOException e) {e.printStackTrace();}
    catch (XPathExpressionException e) {e.printStackTrace();}
}
}
```