JEGYZŐKÖNYV

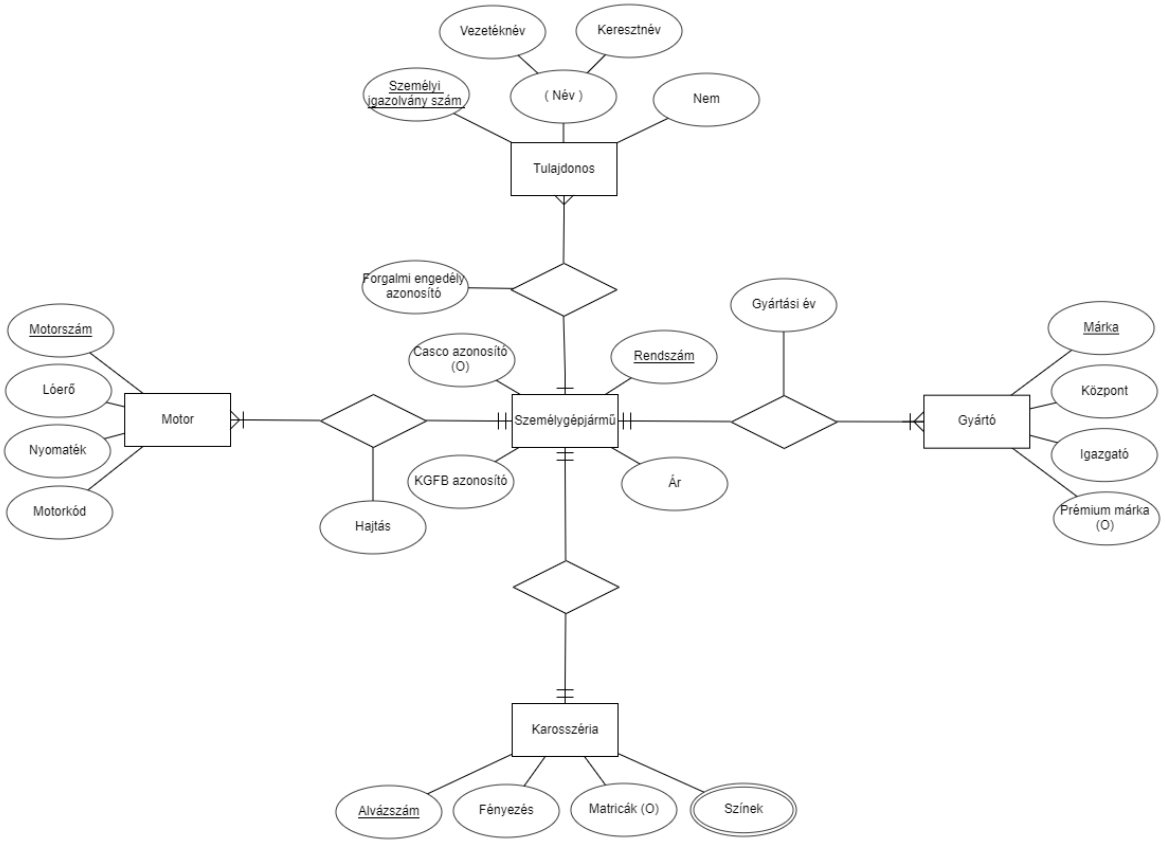
Adatkezelés XML környezetben

KÉSZÍTETTE: FERENCSIK MÁRK
NEPTUN KÓD: UJTWLL

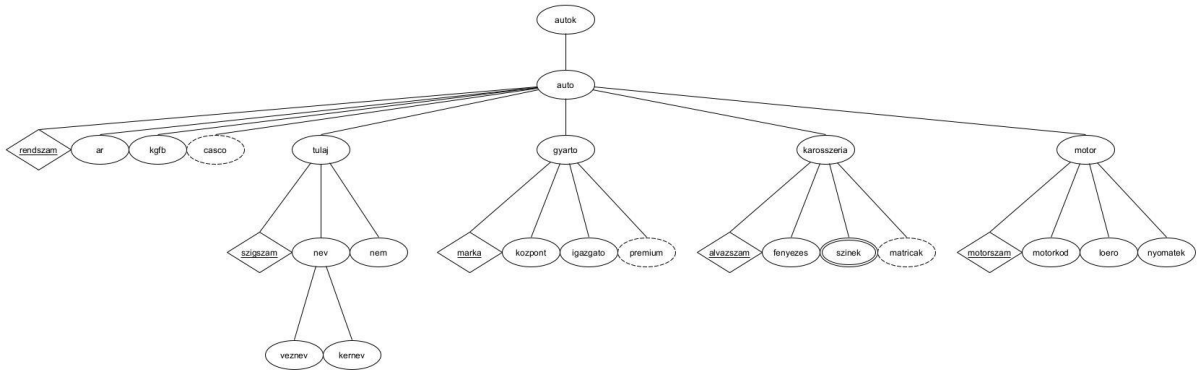
A feladat leírása:

1. feladat:

a. Adatbázis ER modell:



b. Adatbázis konvertálása XDM modellre:



c. XDM modell alapján XML dokumentum készítése:

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

<autok xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="XMLSchemaUJTWLL.xsd">
   <auto rendszam="LFJ873">
      <ar>3000000</ar>
```

```
        <kgfb>87736587</kgfb>
        <tulaj szigszam="893456TA">
           <nev>
              <veznev>Tóth</veznev>
              <kernev>László</kernev>
           </nev>
           <nem>Férfi</nem>
        </tulaj>
        <gyarto marka="Mazda">
           <kozpont>Japán</kozpont>
           <igazgato>Akira Marumoto</igazgato>
           <premium>Nem</premium>
        </gyarto>
        <karosszeria alvazszam="82733649">
           <fenyezes>Matt</fenyezes>
           <szin>Vörös</szin>
           <szin>Fekete</szin>
        </karosszeria>
        <motor motorszam="23347547">
           <motorkod>MR70</motorkod>
           <loero>150</loero>
           <nyomatek>300</nyomatek>
        </motor>
     </auto>
     <auto rendszam="LFJ873">
        <ar>3000000</ar>
        <kgfb>87736587</kgfb>
        <casco>66234891</casco>
        <tulaj szigszam="893456TA">
           <nev>
              <veznev>Tóth</veznev>
              <kernev>László</kernev>
           </nev>
           <nem>Férfi</nem>
        </tulaj>
        <gyarto marka="Mazda">
           <kozpont>Japán</kozpont>
           <igazgato>Akira Marumoto</igazgato>
           <premium>Nem</premium>
        </gyarto>
        <karosszeria alvazszam="82733649">
           <fenyezes>Matt</fenyezes>
           <szin>Vörös</szin>
           <szin>Fekete</szin>
           <matrica>Lángnyelv</matrica>
        </karosszeria>
        <motor motorszam="23347547">
           <motorkod>MR70</motorkod>
           <loero>150</loero>
           <nyomatek>300</nyomatek>
        </motor>
     </auto>
  </autok>
```

d. **XML dokumentum alapján XMLSchema készítése:**

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

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

<xs:simpleType name="arTipus">
   <xs:restriction base="xs:positiveInteger"/>
</xs:simpleType>
```

```
<xs:simpleType name="sorozatszamTipus">
   <xs:restriction base="xs:integer">
      <xs:pattern value="[0-9]{8}"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="nevTipus">
   <xs:restriction base="xs:string">
      <xs:minLength value="2"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="nemTipus">
   <xs:restriction base="xs:string">
      <xs:pattern value="Férfi|férfi|Nő|nő"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="szigszamTipus">
   <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]{6}[A-Z]{2}"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="logikaiTipus">
   <xs:restriction base="xs:string">
      <xs:pattern value="Igen|igen|Nem|nem"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="motorkodTipus">
   <xs:restriction base="xs:string">
      <xs:pattern value="([A-Z0-9])+"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="loeroTipus">
   <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="nyomatekTipus">
   <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
   </xs:restriction>
</xs:simpleType>

<xs:simpleType name="rendszamTipus">
   <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z]{3}[0-9]{3}"/>
   </xs:restriction>
</xs:simpleType>

<xs:element name="autok">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="auto" maxOccurs="unbounded">
            <xs:complexType>
               <xs:sequence>
```

```
                  <xs:element name="ar" type="arTipus"/>
                  <xs:element name="kgfb" type="sorozatszamTipus"/>
                  <xs:element name="casco" type="sorozatszamTipus"
minOccurs="0" maxOccurs="1"/>
                  <xs:element name="tulaj">
                     <xs:complexType>
                        <xs:sequence>
                           <xs:element name="nev">
                              <xs:complexType>
                                 <xs:sequence>
                                    <xs:element name="veznev"
type="nevTipus"/>
                                    <xs:element name="kernev"
type="nevTipus"/>
                                 </xs:sequence>
                              </xs:complexType>
                           </xs:element>
                           <xs:element name="nem" type="nemTipus"/>
                        </xs:sequence>
                        <xs:attribute name="szigszam"
type="szigszamTipus" use="required"/>
                     </xs:complexType>
                  </xs:element>
                  <xs:element name="gyarto">
                     <xs:complexType>
                        <xs:sequence>
                           <xs:element name="kozpont"
type="nevTipus"/>
                           <xs:element name="igazgato"
type="xs:string"/>
                           <xs:element name="premium"
type="logikaiTipus"/>
                        </xs:sequence>
                        <xs:attribute name="marka" type="nevTipus"
use="required"/>
                     </xs:complexType>
                  </xs:element>
                  <xs:element name="karosszeria">
                     <xs:complexType>
                        <xs:sequence>
                           <xs:element name="fenyezes"
type="nevTipus"/>
                           <xs:element name="szin" type="nevTipus"
maxOccurs="unbounded"/>
                           <xs:element name="matrica" type="nevTipus"
minOccurs="0" maxOccurs="1"/>
                        </xs:sequence>
                        <xs:attribute name="alvazszam"
type="sorozatszamTipus" use="required"/>
                     </xs:complexType>
                  </xs:element>
                  <xs:element name="motor">
                     <xs:complexType>
                        <xs:sequence>
                           <xs:element name="motorkod"
type="motorkodTipus"/>
                           <xs:element name="loero"
type="loeroTipus"/>
                           <xs:element name="nyomatek"
type="nyomatekTipus"/>
                        </xs:sequence>
```

```
                        <xs:attribute name="motorszam"
type="sorozatszamTipus" use="required"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="rendszam" type="rendszamTipus"
use="required"/>
        </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>

</xs:schema>
```

2. feladat:

a. **Adatolvasás:**

```
package hu.domparse.UJTWLL;

import java.io.File;
import java.io.IOException;
//import javax.xml.XMLConstants;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
/*import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;*/
import javax.xml.transform.TransformerFactory;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
//import org.xml.sax.SAXParseException;
import org.xml.sax.helpers.DefaultHandler;

public class DOMReadUJTWLL{
    public static void main(String[] args){
        try{
            //Dokumentumolvasó létrehozása
            SAXParserFactory saxParserFactory =
SAXParserFactory.newInstance();

            //SAX értelmező példányosítása, a gyár állítja elő a SAX
elemzőt
            SAXParser saxParser = saxParserFactory.newSAXParser();

            //saját eseménykezelő objektum létrehozása
            SaxHandler handler = new SaxHandler();

            //dokumentum értelmezésének indítása
            saxParser.parse(new File("XMLUJTWLL.xml"), handler);
        } catch (ParserConfigurationException | SAXException |
IOException e){
            e.printStackTrace();
        }
    }
}

//tartalomkezelő keret létrehozása, esemény- és hibakezelő metódusok
definiálása
class SaxHandler extends DefaultHandler{
```

```
    private int indent = 0;

    private String formatAttributes(Attributes attributes){
        int attrLength = attributes.getLength();
        if(attrLength == 0){
            return "";
        }
        StringBuilder sb = new StringBuilder(", {");
        for(int i = 0; i < attrLength; i++){
            sb.append(attributes.getLocalName(i) + ":" +
attributes.getValue(i));
            if(i < attrLength - 1){
                sb.append(", ");
            }
        }
        sb.append("}");
        return sb.toString();
    }

    private void indent(){
        for(int i = 0; i < indent; i++){
            System.out.print("  ");
        }
    }

    //Eseménykezelő metódusok létrehozása, startElement metódus
újraimplementálása
    @Override
    //Elem kezdete és vége
    public void startElement(String uri, String localName, String
qName, Attributes attributes){
        indent++;
        indent();
        System.out.println(qName + formatAttributes(attributes) + "
start");
    }

    @Override //endElement metódust újraimplementáljuk
    public void endElement(String uri, String localName, String
qName){
        indent();
        indent--;
        System.out.println(qName + " end");
    }

    //Szövegelem feldolgozása, characters metódus újraimplementálása
    @Override
    public void characters(char ch[], int start, int length){
        String chars = new String(ch, start, length).trim();
        if(!chars.isEmpty()){
            indent++;
            indent();
            indent--;
            System.out.println(chars);
        }
    }
}
```

b. **Adatlekérdezés:**

```
package hu.domparse.UJTWLL;

import java.io.File;
import javax.xml.parsers.DocumentBuilder;
```

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

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

public class DOMQueryUJTWLL{
    public static void main(String argv[]){
        try{
            File xmlfile = new File("XMLUJTWLL.xml");

            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = factory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlfile);

            doc.getDocumentElement().normalize();

            System.out.println("Root element:
"+doc.getDocumentElement().getNodeName()+"");

            NodeList nList = doc.getElementsByTagName("auto");
            int childCount = nList.getLength();
            System.out.println("Number of child nodes: "+childCount);
            System.out.println("-----------------------------------
");

            for(int i = 0; i < nList.getLength(); i++) {
                Node nNode = nList.item(i);
                System.out.println("Current element: " +
nNode.getNodeName());

                if(nNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element auto = (Element) nNode;
                    String rsz = auto.getAttribute("rendszam");
                    Node kgfbNode =
auto.getElementsByTagName("kgfb").item(0);
                    String kgfb = kgfbNode.getTextContent();
                    Node cascoNode =
auto.getElementsByTagName("casco").item(0);
                    String casco = "";
                    if(cascoNode != null){
                        casco = cascoNode.getTextContent();
                    }
                    Element tulaj = (Element)
auto.getElementsByTagName("tulaj").item(0);
                    String szigszam = tulaj.getAttribute("szigszam");
                    Node veznevNode =
auto.getElementsByTagName("veznev").item(0);
                    String veznev = veznevNode.getTextContent();
                    Node kernevNode =
auto.getElementsByTagName("kernev").item(0);
                    String kernev = kernevNode.getTextContent();
                    Node nemNode =
auto.getElementsByTagName("nem").item(0);
                    String nem = nemNode.getTextContent();
                    Element gyarto = (Element)
auto.getElementsByTagName("gyarto").item(0);
                    String marka = gyarto.getAttribute("marka");
                    Node kozpontNode =
```

```
auto.getElementsByTagName("kozpont").item(0);
                    String kozpont = kozpontNode.getTextContent();
                    Node igazgatoNode =
auto.getElementsByTagName("igazgato").item(0);
                    String igazgato = igazgatoNode.getTextContent();
                    Node premiumNode =
auto.getElementsByTagName("premium").item(0);
                    String premium = premiumNode.getTextContent();
                    Element karosszeria = (Element)
auto.getElementsByTagName("karosszeria").item(0);
                    String alvazszam =
karosszeria.getAttribute("alvazszam");
                    Node fenyezesNode =
auto.getElementsByTagName("fenyezes").item(0);
                    String fenyezes = fenyezesNode.getTextContent();
                    NodeList szinekNode =
auto.getElementsByTagName("szin");
                    String[] szinek = new
String[szinekNode.getLength()];
                    for(int j = 0; j < szinekNode.getLength(); j++){
                        szinek[j] =
szinekNode.item(j).getTextContent();
                    }
                    Node matricaNode =
auto.getElementsByTagName("matrica").item(0);
                    String matrica = "";
                    if(matricaNode != null){
                        matrica = matricaNode.getTextContent();
                    }
                    Element motor = (Element)
auto.getElementsByTagName("motor").item(0);
                    String motorszam =
motor.getAttribute("motorszam");
                    Node motorkodNode =
auto.getElementsByTagName("motorkod").item(0);
                    String motorkod = motorkodNode.getTextContent();
                    Node loeroNode =
auto.getElementsByTagName("loero").item(0);
                    String loero = loeroNode.getTextContent();
                    Node nyomatekNode =
auto.getElementsByTagName("nyomatek").item(0);
                    String nyomatek = nyomatekNode.getTextContent();

                    System.out.printf("Rendszam: %s%n", rsz);
                    System.out.printf("KGFB: %s%n", kgfb);
                    if(cascoNode != null){
                        System.out.printf("Casco: %s%n", casco);;
                    }
                    System.out.printf("Tulajdonos: %18s: %s%n",
"Szemelyi szam", szigszam);
                    System.out.printf("%30s: %s%n", "Vezeteknev",
veznev);
                    System.out.printf("%30s: %s%n", "Keresztnev",
kernev);
                    System.out.printf("%30s: %s%n", "Nem", nem);
                    System.out.printf("Gyarto: %22s: %s%n", "Marka",
marka);
                    System.out.printf("%30s: %s%n", "Kozpont",
kozpont);
                    System.out.printf("%30s: %s%n", "Igazgato",
igazgato);
```

```
                        System.out.printf("%30s: %s%n", "Premium",
premium);
                        System.out.printf("Karosszeria: %17s: %s%n",
"Alvazszam", alvazszam);
                        System.out.printf("%30s: %s%n", "Fenyezes",
fenyezes);
                        for(int k = 0; k < szinekNode.getLength(); k++){
                            System.out.printf("%30s: %s%n", "Szin",
szinek[k]);
                        }
                        if(matricaNode != null){
                            System.out.printf("%30s: %s%n", "Matrica",
matrica);;
                        }
                        System.out.printf("Motor: %23s: %s%n",
"Motorszam", motorszam);
                        System.out.printf("%30s: %s%n", "Motorkod",
motorkod);
                        System.out.printf("%30s: %s%n", "Loero", loero);
                        System.out.printf("%30s: %s%n", "Nyomatek",
nyomatek);
                        System.out.println("----------------------------
--------");
                    }
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
```

c. **Adatmódosítás:**

```
package hu.domparse.UJTWLL;

import java.io.File;
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 org.w3c.dom.*;

public class DOMModifyUJTWLL {
    public static void main(String argv[]) {
        try {
            File inputFile = new File("XMLUJTWLL.xml");
            DocumentBuilderFactory docFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder docBuilder =
docFactory.newDocumentBuilder();
            Document doc = docBuilder.parse(inputFile);

            //1.: A második autó rendszámának megváltoztatása
            Node masodikAuto =
doc.getElementsByTagName("auto").item(1);
            NamedNodeMap attr = masodikAuto.getAttributes();
            Node nodeAttr = attr.getNamedItem("rendszam");
            nodeAttr.setTextContent("PJV931");

            //2.: CASCO eltávolítása az összes autóról
            NodeList autoLista = doc.getElementsByTagName("auto");
```

```
            for(int i = 0; i < autoLista.getLength(); i++){
                Node autoNode = autoLista.item(i);
                NodeList elemLista = autoNode.getChildNodes();
                for(int j = 0; j < elemLista.getLength(); j++){
                    Node elemNode = elemLista.item(j);
                    if(elemNode.getNodeName().equals("casco")){
                        autoNode.removeChild(elemNode);
                    }
                }
            }

            //3.: A matricával rendelkező autókhoz szürke szín
hozzáadása
            NodeList matrica = doc.getElementsByTagName("matrica");
            Node newNode = doc.createElement("szin");
            Text text = doc.createTextNode("Szürke");
            newNode.appendChild(text);
            for(int k = 0; k < matrica.getLength(); k++){
                matrica.item(k).getParentNode().insertBefore(newNode,
matrica.item(k));
            }

            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer =
transformerFactory.newTransformer();
            DOMSource source = new DOMSource(doc);
            System.out.println("-----------Modositott XML-----------
");
            StreamResult consoleResult = new
StreamResult(System.out);
            transformer.transform(source, consoleResult);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```