JEGYZŐKÖNYV

Webes adatkezelő környezetek

Féléves feladat

Autó szervizek

Készítette: **Balogh Dávid**

Neptunkód: **HPQ9EO**

Dátum: **2025. december 1.**

Miskolc, 2025

Tartalomjegyzék

Bevezetés

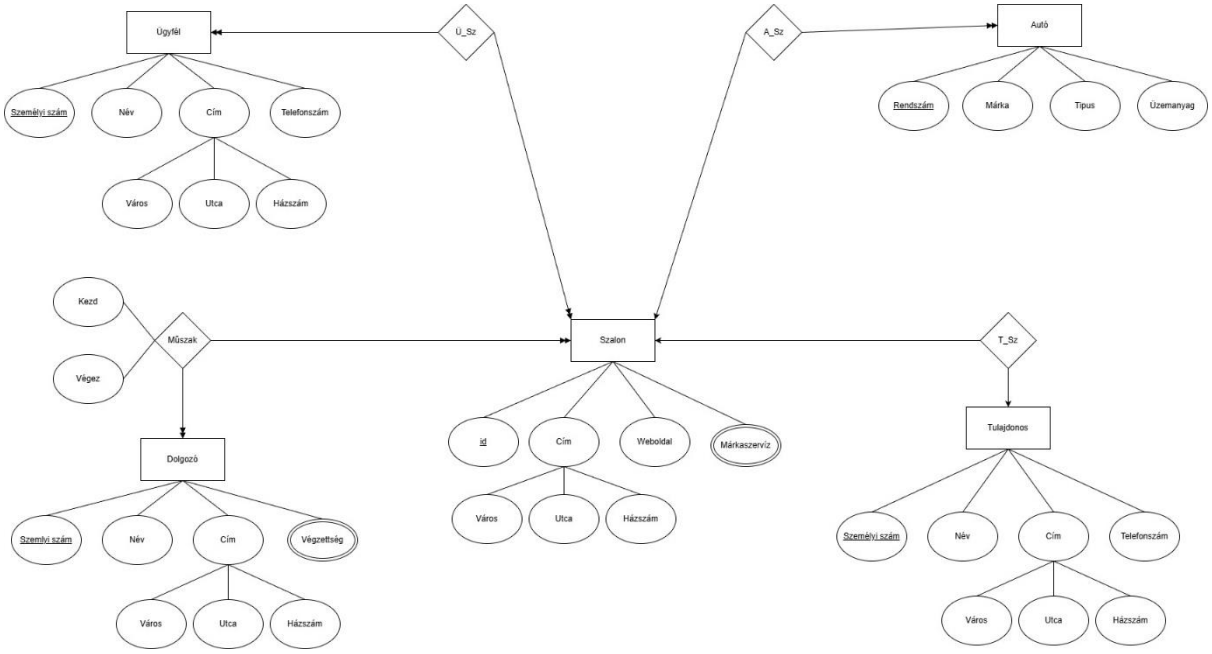
A feladat egy autó szerviz nyilvántartását leíró XML fájl, ahhoz Schema, valamint feldolgozó Java program írása.

A feladat leírása

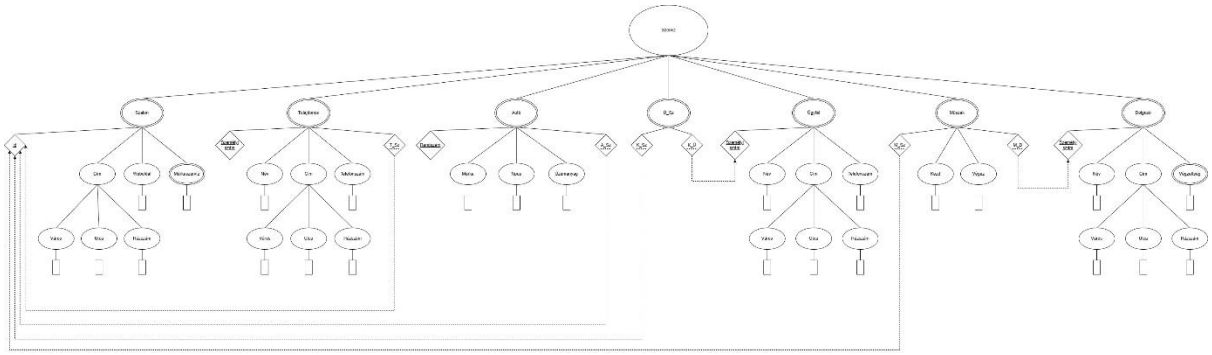
A nyilvántartásban szalonokat tárolunk, minden szalonnak van 1 tulajdonosa és a tulajdonosoknak is csak 1 szalonja lehet. A tulajdonosok azonosítására a személyi számot használjuk, valamint tároljuk, hogy melyik szalon az övék. Ezen kívül a nevét címét és telefonszámát tároljuk. A szalonoknak az azonosítására egy id-t használunk, ezen kívül tároljuk a címet weboldal linket, valamint a szerződött márkaszervizeket. Ezeken kívül tároljuk az autókat, amiket rendszám alapján azonosítunk és minden autóhoz hozzárendelünk egy szalont. Ezen kívül tároljuk még a márkát, típust és hogy milyen üzemanyag kell bele. Az autók és a szalonok egy-több kapcsolatban állnak egymással. Továbbá tároljuk az egyes szalonokba látogató ügyfeleket is, akiket szintén személyi szám alapján azonosítjuk, valamint tároljuk a nevüket, címüket és telefonszámukat. Mivel az ügyfelek és a szalonok több-több kapcsolatban állnak így tárolnunk kell a kapcsolatot is köztük, ami csak a két azonosítót tartalmazza. A szalonokban természetesen vannak dolgozók is, akiket szintén nyilván kell tartani, őket szintén személyi szám alapján azonosítjuk. Tároljuk még a nevüket, címüket és végzettségeiket. A dolgozóknak a műszakjait is szeretnénk tárolni, valamint egyes emberek dolgozhatnak akár több szalonban is, szóval a dolgozók és a szalonok között szintén több-több kapcsolat van, ami most tartalmazza az azonosítókon kívül a műszak kezdetét és végét.

1. Szerviz XML elkészítése Schema-val a modellek alapján

1.1 Az adatbázis ER modell tervezése



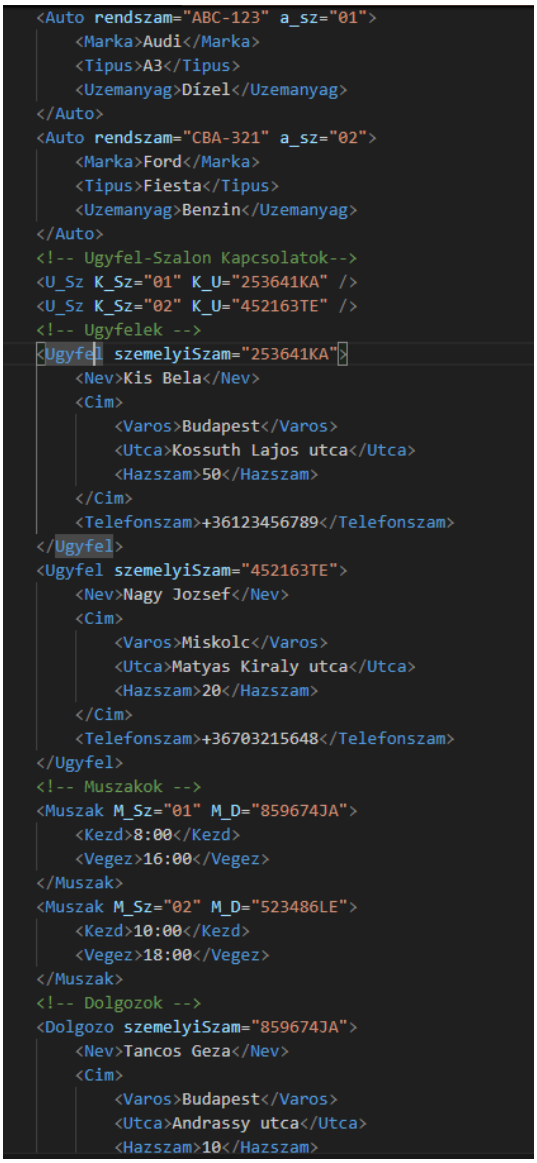
1.2 Az adatbázis konvertálása XDM modellre



1.3 Az XDM modell alapján XML dokumentum készítése:

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

<Szerviz xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:noNamespaceSchemaLocation="HPQ9EO_XMLSchema.xsd">
    <!-- Tulajdonosok -->
    <Tulajdonos szemelyiSzam="123456NA" t_sz="01">
        <Nev>Kovács János</Nev>
        <Cim>
            <Varos>Budapest</Varos>
            <Utca>Kossuth Lajos utca</Utca>
            <Hazszam>10</Hazszam>
        </Cim>
        <Telefonszam>+363025815235</Telefonszam>
    </Tulajdonos>
    <Tulajdonos szemelyiSzam="987654TA" t_sz="02">
        <Nev>Remek Elek</Nev>
        <Cim>
            <Varos>Miskolc</Varos>
            <Utca>Kiraly utca</Utca>
            <Hazszam>5</Hazszam>
        </Cim>
        <Telefonszam>+36706584123</Telefonszam>
    </Tulajdonos>
    <!-- Szalonok -->
    <Szalon id="01">
        <Cim>
            <Varos>Budapest</Varos>
            <Utca>Fo utca</Utca>
            <Hazszam>1</Hazszam>
        </Cim>
        <Weboldal>www.budapestauto.hu</Weboldal>
        <Markaszerviz>Audi</Markaszerviz>
    </Szalon>
    <Szalon id="02">
        <Cim>
            <Varos>Miskolc</Varos>
            <Utca>Derine utca</Utca>
            <Hazszam>7</Hazszam>
        </Cim>
        <Weboldal>www.miskolcauto.hu</Weboldal>
        <Markaszerviz>Mercedes</Markaszerviz>
        <Markaszerviz>Ford</Markaszerviz>
    </Szalon>
```



```xml
    <Auto rendszam="ABC-123" a_sz="01">
        <Marka>Audi</Marka>
        <Tipus>A3</Tipus>
        <Uzemanyag>Dízel</Uzemanyag>
    </Auto>
    <Auto rendszam="CBA-321" a_sz="02">
        <Marka>Ford</Marka>
        <Tipus>Fiesta</Tipus>
        <Uzemanyag>Benzin</Uzemanyag>
    </Auto>
    <!-- Ugyfel-Szalon Kapcsolatok-->
    <U_Sz K_Sz="01" K_U="253641KA" />
    <U_Sz K_Sz="02" K_U="452163TE" />
    <!-- Ugyfelek -->
    <Ugyfel szemelyiSzam="253641KA">
        <Nev>Kis Bela</Nev>
        <Cim>
            <Varos>Budapest</Varos>
            <Utca>Kossuth Lajos utca</Utca>
            <Hazszam>50</Hazszam>
        </Cim>
        <Telefonszam>+36123456789</Telefonszam>
    </Ugyfel>
    <Ugyfel szemelyiSzam="452163TE">
        <Nev>Nagy Jozsef</Nev>
        <Cim>
            <Varos>Miskolc</Varos>
            <Utca>Matyas Kiraly utca</Utca>
            <Hazszam>20</Hazszam>
        </Cim>
        <Telefonszam>+36703215648</Telefonszam>
    </Ugyfel>
    <!-- Muszakok -->
    <Muszak M_Sz="01" M_D="859674JA">
        <Kezd>8:00</Kezd>
        <Vegez>16:00</Vegez>
    </Muszak>
    <Muszak M_Sz="02" M_D="523486LE">
        <Kezd>10:00</Kezd>
        <Vegez>18:00</Vegez>
    </Muszak>
    <!-- Dolgozok -->
    <Dolgozo szemelyiSzam="859674JA">
        <Nev>Tancos Geza</Nev>
        <Cim>
            <Varos>Budapest</Varos>
            <Utca>Andrassy utca</Utca>
            <Hazszam>10</Hazszam>
```

1.4 Az XML dokumentum alapján XMLSchema készítése

A Schema szekvenciálisan tartalmazza az egyes elemeket, saját típusokkal és végtelen előfordulással. Továbbá tartalmazza, hogy melyik attribútumok a kulcsok, idegen kulcsok és azok melyik azonosítóra hivatkoznak. Valamint az 1-1 kapcsolat unique mező segítségével van megvalósítva. A tulajdonos, szalon, autó, kapcsolat, ügyfél, műszak és dolgozó típuson kívül van egy cím típus is ugyanis az minden előfordulásánál egy összetett tulajdonság, ami tartalmazza a várost, utcát és a házszámot.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <!-- Egyszerű típusok -->
    <xs:element name="Szerviz">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Tulajdonos" type="tulajTipus" maxOccurs="unbounded" />
                <xs:element name="Szalon" type="szalonTipus" maxOccurs="unbounded" />
                <xs:element name="Auto" type="autoTipus" maxOccurs="unbounded" />
                <xs:element name="U_Sz" type="kapcsolatTipus" maxOccurs="unbounded" />
                <xs:element name="Ugyfel" type="ugyfelTipus" maxOccurs="unbounded" />
                <xs:element name="Muszak" type="muszakTipus" maxOccurs="unbounded" />
                <xs:element name="Dolgozo" type="dolgozoTipus" maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>

        <!-- Kulcsok -->
        <xs:key name="tulaj_kulcs">
            <xs:selector xpath="Tulajdonos" />
            <xs:field xpath="@szemelyiSzam" />
        </xs:key>
        <xs:key name="szalon_kulcs">
            <xs:selector xpath="Szalon" />
            <xs:field xpath="@id" />
        </xs:key>
        <xs:key name="auto_kulcs">
            <xs:selector xpath="Auto" />
            <xs:field xpath="@rendszam" />
        </xs:key>
        <xs:key name="ugyfel_kulcs">
            <xs:selector xpath="Ugyfel" />
            <xs:field xpath="@szemelyiSzam" />
        </xs:key>
        <xs:key name="dolgozo_kulcs">
            <xs:selector xpath="Dolgozo" />
            <xs:field xpath="@szemelyiSzam" />
        </xs:key>
        <!-- Idegen Kulcsok -->
        <xs:keyref refer="szalon_kulcs" name="szalon_idegen_kulcs">
            <xs:selector xpath="Muszak" />
            <xs:field xpath="@M_Sz" />
        </xs:keyref>
        <xs:keyref refer="dolgozo_kulcs" name="dolgozo_idegen_kulcs">
            <xs:selector xpath="Muszak" />
            <xs:field xpath="@M_D" />
        </xs:keyref>
```

```
    <xs:keyref refer="szalon_kulcs" name="szalon_idegen_kulcs2">
        <xs:selector xpath="U_Sz" />
        <xs:field xpath="@K_Sz" />
    </xs:keyref>
    <xs:keyref refer="ugyfel_kulcs" name="ugyfel_idegen_kulcs">
        <xs:selector xpath="U_Sz" />
        <xs:field xpath="@K_U" />
    </xs:keyref>
    <xs:keyref refer="szalon_kulcs" name="auto_idegen_kulcs">
        <xs:selector xpath="Auto" />
        <xs:field xpath="@a_sz"></xs:field>
    </xs:keyref>
    <!-- Az 1:1 kapcsolat megvalósításához -->
    <xs:unique name="unique_tulaj">
        <xs:selector xpath="Tulajdonos" />
        <xs:field xpath="@t_sz" />
    </xs:unique>
</xs:element>

<!-- Komplex Tipusok -->
<xs:complexType name="tulajTipus">
    <xs:sequence>
        <xs:element name="Nev" type="xs:string" />
        <xs:element name="Cim" type="cimTipus" />
        <xs:element name="Telefonszam" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="szemelyiSzam" type="xs:string" use="required" />
    <xs:attribute name="t_sz" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="szalonTipus">
    <xs:sequence>
        <xs:element name="Cim" type="cimTipus" />
        <xs:element name="Weboldal" type="xs:string" />
        <xs:element name="Markaszerviz" type="xs:string" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="autoTipus">
    <xs:sequence>
        <xs:element name="Marka" type="xs:string" />
        <xs:element name="Tipus" type="xs:string" />
        <xs:element name="Uzemanyag" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="rendszam" type="xs:string" use="required" />
    <xs:attribute name="a_sz" type="xs:string" use="required" />
```

```
    <xs:complexType name="kapcsolatTipus">
        <xs:attribute name="K_Sz" type="xs:string" use="required" />
        <xs:attribute name="K_U" type="xs:string" use="required" />
    </xs:complexType>
    <xs:complexType name="ugyfelTipus">
        <xs:sequence>
            <xs:element name="Nev" type="xs:string" />
            <xs:element name="Cim" type="cimTipus" />
            <xs:element name="Telefonszam" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="szemelyiSzam" type="xs:string" use="required" />
    </xs:complexType>
    <xs:complexType name="muszakTipus">
        <xs:sequence>
            <xs:element name="Kezd" type="xs:string" />
            <xs:element name="Vegez" type="xs:string" />
        </xs:sequence>
        <xs:attribute name="M_Sz" type="xs:string" use="required" />
        <xs:attribute name="M_D" type="xs:string" use="required" />
    </xs:complexType>
    <xs:complexType name="dolgozoTipus">
        <xs:sequence>
            <xs:element name="Nev" type="xs:string" />
            <xs:element name="Cim" type="cimTipus" />
            <xs:element name="Vegzettseg" type="xs:string" maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="szemelyiSzam" type="xs:string" use="required" />
    </xs:complexType>
    <xs:complexType name="cimTipus">
        <xs:sequence>
            <xs:element name="Varos" type="xs:string" />
            <xs:element name="Utca" type="xs:string" />
            <xs:element name="Hazszam" type="xs:integer" />
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

2. Szerviz XML feldolgozása Java programban

2.1 adatolvasás

Az olvasó osztály megkeresi és beolvassa a létrehozott XML fájlt feldolgozza és kiírja mind a konzolra, mind az output_read.txt fájlba. Az XML feldolgozásához írtam egy függvényt, ami minden rész előtt kiírja, hogy éppen tulajdonosok, szalonok, vagy mások adatait fogjuk látni és utána megjeleníti az adott csoport tulajdonságait.

```java
private static void processElements(Document doc, String tagName, String sectionName, FileWriter writer)
        throws Exception {
    System.out.println("\n=== " + sectionName + " ===");
    writer.write("\n=== " + sectionName + " ===\n");

    NodeList nodeList = doc.getElementsByTagName(tagName);
    for (int i = 0; i < nodeList.getLength(); i++) {
        Node node = nodeList.item(i);

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;
            System.out.println("\n" + tagName + " " + (i + 1) + ":");
            writer.write("\n" + tagName + " " + (i + 1) + ":\n");

            // Attribútumok kiírása
            if (element.hasAttributes()) {
                for (int j = 0; j < element.getAttributes().getLength(); j++) {
                    Node attr = element.getAttributes().item(j);
                    System.out.println("  " + attr.getNodeName() + ": " + attr.getNodeValue());
                    writer.write("  " + attr.getNodeName() + ": " + attr.getNodeValue() + "\n");
                }
            }

            // Gyermek elemek kiírása
            NodeList children = element.getChildNodes();
            for (int j = 0; j < children.getLength(); j++) {
                Node child = children.item(j);
                if (child.getNodeType() == Node.ELEMENT_NODE) {
                    String childName = child.getNodeName();
                    String childValue = child.getTextContent();
                    System.out.println("  " + childName + ": " + childValue);
                    writer.write("  " + childName + ": " + childValue + "\n");
                }
            }
        }
    }
    System.out.println(x: "--------------------------------");
    writer.write(str: "--------------------------------\n");
}
```

2.2 adatírás

Hasonlóan, mint az olvasó osztály, ez is megkeresi az XML-t beolvassa és kiírja a konzolra, valamint létrehoz egy új XML fájlt. Ennek megvalósítását is kiszerveztem függvényekbe a jobb olvashatóság és rövidebb kód érdekében.

```java
// Rekurzív metódus a dokumentum fa struktúrájának kiírására
private static void printDocumentTree(Node node, int depth) {
    if (node.getNodeType() == Node.ELEMENT_NODE) {
        // Behúzás a mélység alapján
        StringBuilder indent = new StringBuilder();
        for (int i = 0; i < depth; i++) {
            indent.append(str: "  ");
        }

        // Elem nyitó tagje attribútumokkal
        Element element = (Element) node;
        System.out.println(indent + "<" + element.getNodeName() + getAttributes(element) + ">");

        // Szöveges tartalom ellenőrzése
        boolean hasElementChildren = hasElementChildren(element);
        String textContent = element.getTextContent().trim();

        // Ha van szöveges tartalom és nincs element gyermek, kiírjuk
        if (!textContent.isEmpty() && !hasElementChildren) {
            System.out.println(indent + "  " + textContent);
        }

        // Gyermek elemek feldolgozása
        NodeList children = element.getChildNodes();
        for (int i = 0; i < children.getLength(); i++) {
            printDocumentTree(children.item(i), depth + 1);
        }

        // Elem záró tagje (csak ha van element gyermek vagy a szöveg nem üres)
        if (hasElementChildren || (!textContent.isEmpty() && hasElementChildren)) {
            System.out.println(indent + "</" + element.getNodeName() + ">");
        }

    } else if (node.getNodeType() == Node.TEXT_NODE) {
        // Szöveges tartalom kiírása, ha nem üres
        String text = node.getTextContent().trim();
        if (!text.isEmpty()) {
            StringBuilder indent = new StringBuilder();
            for (int i = 0; i < depth; i++) {
                indent.append(str: "  ");
            }
            System.out.println(indent + text);
        }
    }
```

```java
// Ellenőrzi, hogy egy elemnek van-e element típusú gyermeke
private static boolean hasElementChildren(Element element) {
    NodeList children = element.getChildNodes();
    for (int i = 0; i < children.getLength(); i++) {
        if (children.item(i).getNodeType() == Node.ELEMENT_NODE) {
            return true;
        }
    }
    return false;
}

// Az elem attribútumainak összegyűjtése string formátumba
private static String getAttributes(Element element) {
    StringBuilder attributes = new StringBuilder();
    if (element.hasAttributes()) {
        for (int i = 0; i < element.getAttributes().getLength(); i++) {
            Node attr = element.getAttributes().item(i);
            attributes.append(str: " ").append(attr.getNodeName())
                    .append(str: "=\"").append(attr.getNodeValue()).append(str: "\"");
        }
    }
    return attributes.toString();
}
```

```java
// Dokumentum mentése fájlba
private static void saveDocumentToFile(Document doc, String filename) {
    try {
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();

        // Formázási beállítások
        transformer.setOutputProperty(javax.xml.transform.OutputKeys.INDENT, value: "yes");
        transformer.setOutputProperty(javax.xml.transform.OutputKeys.ENCODING, value: "UTF-8");
        transformer.setOutputProperty(name: "{http://xml.apache.org/xslt}indent-amount", value: "4");

        DOMSource source = new DOMSource(doc);
        StreamResult result = new StreamResult(new File(filename));
        transformer.transform(source, result);

    } catch (Exception e) {
        System.err.println("Hiba a fájl mentése során: " + e.getMessage());
        e.printStackTrace();
    }
}
```

2.3 adat lekérdezés

Ez az osztály a beolvasott XML fájlból csinál 5 különböző lekérdezést.

- Audi márkájú autók kilistázása
- Budapesti szalonok kilistázása
- Autószerelő végzetséggel rendelkező dolgozók kiírása
- Dízeles autók megjelenítése
- Miskolci címmel rendelkező tulajdonosok listázása

Ezeket a lekérdezéseket is kiszerveztem külön függvényekbe, mint ahogy az előző osztályokban is tettem.

```java
// 1. Lekérdezés: Audi autók
private static void queryAudiAutos(Document doc) {
    NodeList autoList = doc.getElementsByTagName(tagname: "Auto");
    for (int i = 0; i < autoList.getLength(); i++) {
        Element auto = (Element) autoList.item(i);
        String marka = auto.getElementsByTagName(name: "Marka").item(index: 0).getTextContent();
        if ("Audi".equals(marka)) {
            String rendszam = auto.getAttribute(name: "rendszam");
            String tipus = auto.getElementsByTagName(name: "Tipus").item(index: 0).getTextContent();
            System.out.println("  - " + marka + " " + tipus + " (" + rendszam + ")");
        }
    }
}

// 2. Lekérdezés: Budapesti szalonok
private static void queryBudapestSzalonok(Document doc) {
    NodeList szalonList = doc.getElementsByTagName(tagname: "Szalon");
    for (int i = 0; i < szalonList.getLength(); i++) {
        Element szalon = (Element) szalonList.item(i);
        Element cim = (Element) szalon.getElementsByTagName(name: "Cim").item(index: 0);
        String varos = cim.getElementsByTagName(name: "Varos").item(index: 0).getTextContent();

        if ("Budapest".equals(varos)) {
            String id = szalon.getAttribute(name: "id");
            String weboldal = szalon.getElementsByTagName(name: "Weboldal").item(index: 0).getTextContent();
            System.out.println("  - Szalon ID: " + id + ", Web: " + weboldal);

            // Márkaszervizek listázása
            NodeList markak = szalon.getElementsByTagName(name: "Markaszerviz");
            System.out.print(s: "    Márkák: ");
            for (int j = 0; j < markak.getLength(); j++) {
                System.out.print(markak.item(j).getTextContent());
                if (j < markak.getLength() - 1)
                    System.out.print(s: ", ");
            }
            System.out.println();
        }
    }
}
```

```java
// 3. Lekérdezés: Autószerelő végzettségű dolgozók
private static void queryAutoszereloDolgozok(Document doc) {
    NodeList dolgozoList = doc.getElementsByTagName(tagname: "Dolgozo");
    for (int i = 0; i < dolgozoList.getLength(); i++) {
        Element dolgozo = (Element) dolgozoList.item(i);
        NodeList vegzettsegek = dolgozo.getElementsByTagName(name: "Vegzettseg");
        boolean autoszerelo = false;

        for (int j = 0; j < vegzettsegek.getLength(); j++) {
            if ("Autoszerelo".equals(vegzettsegek.item(j).getTextContent())) {
                autoszerelo = true;
                break;
            }
        }

        if (autoszerelo) {
            String nev = dolgozo.getElementsByTagName(name: "Nev").item(index: 0).getTextContent();
            String szemelyiSzam = dolgozo.getAttribute(name: "szemelyiSzam");
            System.out.println("  - " + nev + " (" + szemelyiSzam + ")");
        }
    }
}

// 4. Lekérdezés: Dízel üzemanyagú autók
private static void queryDieselAutos(Document doc) {
    NodeList autoList = doc.getElementsByTagName(tagname: "Auto");
    for (int i = 0; i < autoList.getLength(); i++) {
        Element auto = (Element) autoList.item(i);
        String uzemanyag = auto.getElementsByTagName(name: "Uzemanyag").item(index: 0).getTextContent();
        if ("Dízel".equals(uzemanyag)) {
            String marka = auto.getElementsByTagName(name: "Marka").item(index: 0).getTextContent();
            String tipus = auto.getElementsByTagName(name: "Tipus").item(index: 0).getTextContent();
            String rendszam = auto.getAttribute(name: "rendszam");
            System.out.println("  - " + marka + " " + tipus + " (" + rendszam + ") - " + uzemanyag);
        }
    }
}
```

```java
// 5. Lekérdezés: Miskolci című tulajdonosok
private static void queryMiskolciTulajdonosok(Document doc) {
    NodeList tulajdonosList = doc.getElementsByTagName(tagname: "Tulajdonos");
    for (int i = 0; i < tulajdonosList.getLength(); i++) {
        Element tulajdonos = (Element) tulajdonosList.item(i);
        Element cim = (Element) tulajdonos.getElementsByTagName(name: "Cim").item(index: 0);
        String varos = cim.getElementsByTagName(name: "Varos").item(index: 0).getTextContent();

        if ("Miskolc".equals(varos)) {
            String nev = tulajdonos.getElementsByTagName(name: "Nev").item(index: 0).getTextContent();
            String telefonszam = tulajdonos.getElementsByTagName(name: "Telefonszam").item(index: 0).getTextContent();
            System.out.println("  - " + nev + ", Tel: " + telefonszam);
        }
    }
}
```

2.4 adatmódosítás

Az előző részfeladatokkal megegyezően, ez az osztály is beolvassa az XML-t és végrehajt 6 módosítást rajta majd ellenőrzi, hogy a módosítások után is valid-e az fájl és elmenti egy másik XML fájlba.

Módosítások:

- Audi A3 megváltoztatás A4-re
- Egyik tulajdonos telefonszámának módosítása
- Egyik dolgozóhoz új végzettség felvitele
- Új márkaszerviz felvétele az egyik szalonhoz

- Az egyik műszak időpontjának módosítása
- Új ügyfél-szalon kapcsolat hozzáadása

Az előző osztályokhoz hasonlóan itt is kiszerveztem a módosításokat és a validitás ellenőrzését külön függvényekbe.

```java
// 1. Módosítás: Audi A3 típus módosítása A4-re
private static void modifyAudiType(Document doc) {
    NodeList autoList = doc.getElementsByTagName(tagname: "Auto");
    for (int i = 0; i < autoList.getLength(); i++) {
        Element auto = (Element) autoList.item(i);
        String marka = auto.getElementsByTagName(name: "Marka").item(index: 0).getTextContent();
        String tipus = auto.getElementsByTagName(name: "Tipus").item(index: 0).getTextContent();

        if ("Audi".equals(marka) && "A3".equals(tipus)) {
            String rendszam = auto.getAttribute(name: "rendszam");
            System.out.println("  - Módosítás előtt: " + marka + " " + tipus + " (" + rendszam + ")");
            auto.getElementsByTagName(name: "Tipus").item(index: 0).setTextContent(textContent: "A4");
            System.out.println("  - Módosítás után: " + marka + " A4 (" + rendszam + ")");
            break;
        }
    }
}

// 2. Módosítás: Telefonszám módosítása
private static void modifyPhoneNumber(Document doc) {
    NodeList tulajdonosList = doc.getElementsByTagName(tagname: "Tulajdonos");
    for (int i = 0; i < tulajdonosList.getLength(); i++) {
        Element tulajdonos = (Element) tulajdonosList.item(i);
        String nev = tulajdonos.getElementsByTagName(name: "Nev").item(index: 0).getTextContent();

        if ("Kovács János".equals(nev)) {
            String regiTelefon = tulajdonos.getElementsByTagName(name: "Telefonszam").item(index: 0).getTextContent();
            System.out.println("  - " + nev + " régi telefonszáma: " + regiTelefon);
            tulajdonos.getElementsByTagName(name: "Telefonszam").item(index: 0).setTextContent(textContent: "+36301112233");
            String ujTelefon = tulajdonos.getElementsByTagName(name: "Telefonszam").item(index: 0).getTextContent();
            System.out.println("  - " + nev + " új telefonszáma: " + ujTelefon);
            break;
        }
    }
}
```

```java
// 3. Módosítás: Dolgozó végzettség hozzáadása
private static void addVegzettsegToDolgozo(Document doc) {
    NodeList dolgozoList = doc.getElementsByTagName(tagname: "Dolgozo");
    for (int i = 0; i < dolgozoList.getLength(); i++) {
        Element dolgozo = (Element) dolgozoList.item(i);
        String nev = dolgozo.getElementsByTagName(name: "Nev").item(index: 0).getTextContent();

        if ("Tancos Geza".equals(nev)) {
            Element ujVegzettseg = doc.createElement(tagName: "Vegzettseg");
            ujVegzettseg.appendChild(doc.createTextNode(data: "Műszaki ellenőr"));
            dolgozo.appendChild(ujVegzettseg);

            System.out.println("  - " + nev + " új végzettsége: Műszaki ellenőr");

            // Kiírjuk az összes végzettséget
            NodeList vegzettsegek = dolgozo.getElementsByTagName(name: "Vegzettseg");
            System.out.print(s: "    Összes végzettsége: ");
            for (int j = 0; j < vegzettsegek.getLength(); j++) {
                System.out.print(vegzettsegek.item(j).getTextContent());
                if (j < vegzettsegek.getLength() - 1)
                    System.out.print(s: ", ");
            }
            System.out.println();
            break;
        }
    }
}
```

```java
// 4. Módosítás: Új márkaszerviz hozzáadása szalonhoz
private static void addNewMarkaToSzalon(Document doc) {
    NodeList szalonList = doc.getElementsByTagName(tagname: "Szalon");
    for (int i = 0; i < szalonList.getLength(); i++) {
        Element szalon = (Element) szalonList.item(i);
        String id = szalon.getAttribute(name: "id");

        if ("02".equals(id)) { // Miskolci szalon
            Element ujMarka = doc.createElement(tagName: "Markaszerviz");
            ujMarka.appendChild(doc.createTextNode(data: "BMW"));
            szalon.appendChild(ujMarka);

            System.out.println("  - Szalon ID " + id + " új márkája: BMW");

            // Kiírjuk az összes márkaszervizt
            NodeList markak = szalon.getElementsByTagName(name: "Markaszerviz");
            System.out.print(s: "    Összes márkája: ");
            for (int j = 0; j < markak.getLength(); j++) {
                System.out.print(markak.item(j).getTextContent());
                if (j < markak.getLength() - 1)
                    System.out.print(s: ", ");
            }
            System.out.println();
            break;
        }
    }
}
```

```java
// 5. Módosítás: Műszak időpontjának módosítása
private static void modifyMuszakIdopont(Document doc) {
    NodeList muszakList = doc.getElementsByTagName(tagname: "Muszak");
    for (int i = 0; i < muszakList.getLength(); i++) {
        Element muszak = (Element) muszakList.item(i);
        String mSz = muszak.getAttribute(name: "M_Sz");

        if ("01".equals(mSz)) {
            String regiKezd = muszak.getElementsByTagName(name: "Kezd").item(index: 0).getTextContent();
            String regiVegez = muszak.getElementsByTagName(name: "Vegez").item(index: 0).getTextContent();

            System.out.println("  - Műszak " + mSz + " régi időpontja: " + regiKezd + " - " + regiVegez);

            muszak.getElementsByTagName(name: "Kezd").item(index: 0).setTextContent(textContent: "9:00");
            muszak.getElementsByTagName(name: "Vegez").item(index: 0).setTextContent(textContent: "17:00");

            String ujKezd = muszak.getElementsByTagName(name: "Kezd").item(index: 0).getTextContent();
            String ujVegez = muszak.getElementsByTagName(name: "Vegez").item(index: 0).getTextContent();
            System.out.println("  - Műszak " + mSz + " új időpontja: " + ujKezd + " - " + ujVegez);
            break;
        }
    }
}
```

```java
// 6. Módosítás: Új kapcsolat hozzáadása
private static void addNewKapcsolat(Document doc) {
    Element root = doc.getDocumentElement();

    // Új kapcsolat elem létrehozása - csak létező kulcsokkal
    Element ujKapcsolat = doc.createElement(tagName: "U_Sz");
    ujKapcsolat.setAttribute(name: "K_Sz", value: "01"); // Létező szalon ID
    ujKapcsolat.setAttribute(name: "K_U", value: "253641KA"); // Létező ügyfél személyi szám

    // H
    Node NodeList uSzList - hpq9eo.domparse.hu.HPQ9EODomModify.addNewKapcsolat(Document)
    if (uSzList.getLength() > 0) {
        // Az utolsó U_Sz elem után szúrjuk be
        Node lastUSz = uSzList.item(uSzList.getLength() - 1);
        root.insertBefore(ujKapcsolat, lastUSz.getNextSibling());
    } else {
        // Ha nincs U_Sz elem, akkor az Autók után, Ügyfelek előtt
        NodeList autoList = doc.getElementsByTagName(tagname: "Auto");
        if (autoList.getLength() > 0) {
            Node lastAuto = autoList.item(autoList.getLength() - 1);
            root.insertBefore(ujKapcsolat, lastAuto.getNextSibling());
        }
    }

    System.out.println(x: "  - Új kapcsolat hozzáadva: Szalon ID 01 - Ügyfél 253641KA");
}
```

```java
// Séma validálás ellenőrzése
private static boolean validateAgainstSchema(Document doc) {
    try {
        // A séma validálás bekapcsolása
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        dbFactory.setNamespaceAware(awareness: true);
        dbFactory.setValidating(validating: true);
        dbFactory.setAttribute(name: "http://java.sun.com/xml/jaxp/properties/schemaLanguage",
                value: "http://www.w3.org/2001/XMLSchema");

        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

        // Error handler a validációs hibák kezelésére
        dBuilder.setErrorHandler(new org.xml.sax.ErrorHandler() {
            public void warning(org.xml.sax.SAXParseException e) throws org.xml.sax.SAXException {
                System.out.println("Validálási figyelmeztetés: " + e.getMessage());
            }

            public void error(org.xml.sax.SAXParseException e) throws org.xml.sax.SAXException {
                System.out.println("Validálási hiba: " + e.getMessage());
                throw e;
            }

            public void fatalError(org.xml.sax.SAXParseException e) throws org.xml.sax.SAXException {
                System.out.println("Validálási kritikus hiba: " + e.getMessage());
                throw e;
            }
        });

        // Újra betöltjük és validáljuk a módosított dokumentumot
        File modifiedFile = new File(pathname: "HPQ9EO_modified.xml");
        dBuilder.parse(modifiedFile);
        return true;

    } catch (Exception e) {
        System.out.println("Validálási hiba: " + e.getMessage());
        return false;
    }
}
```