JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

XMLTaskC7H5VB

Készítette: **Csicsely Gábor**

Neptunkód: **C7H5VB**

Dátum: 2023.12.03

Tartalomjegyzék

A feladat leírása:

A feladatban a Hell cég működésével kapcsolatos ER modellből készítettem egy XDM modellt, amiből készült egy XML és XSD valamint hozzá egy azt kezelő DOM program az Eclipse IDE használatával.

Az XSD és XML validálásához egy online XML validátor volt használva.

A rendszer nyilvántartja a Dolgozókat(ID, Név, Beosztás, Fizetés, Telefonszám, Lakcím), a Hell termékeket gyártó Gyárakat(Gyár Azonosító, Gyártási ráta, Kiadások, Címe) a Termékeket(Vállalati Azonosító, Fajtája, Bevétel, Kiadások) amiket gyártanak.
Ezen felül nyilván vannak tartva a Megrendelők(Megrendelő Azonosító, Elérhetőség, Címe) és a Hell cégnek egyik legnagyobb látványossága, üdülő központja, az Avalon Park(Azonosító, Bevétel, Kiadások, Címe) és annak Vezetője(ID, Név, Lakcím)

A rendszer megbízhatóan nyilván tudja tartani így a bevételeket, fizetéseket, kiadásokat, valamint a gyárak gyártási rátáját, azt, hogy melyik megrendelő mennyit, és melyik gyárból rendel.

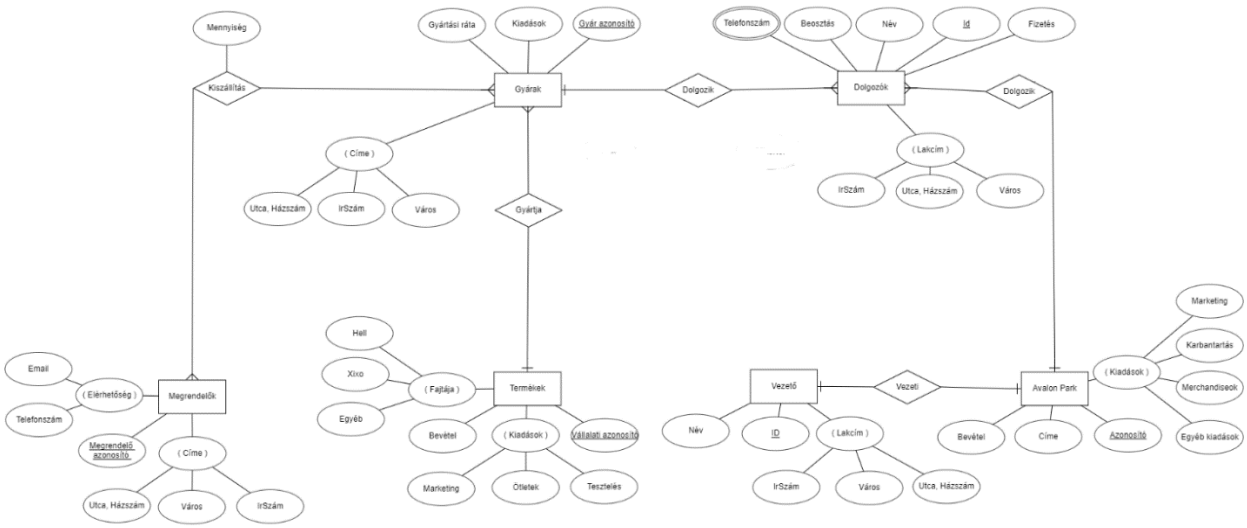
Az XML dokumentum szabályosságára egy hozzá írt XSD dokumentum figyel, ezzel biztosítva a hibátlan működést.

A hozzá írt Java DOM program képes egyszerűbb lekérdezésekre, az XML kiírására, valamint módosítására és annak mentésére.

1. feladat

1a) Az adatbázis ER modell tervezése

Az ER Modell:



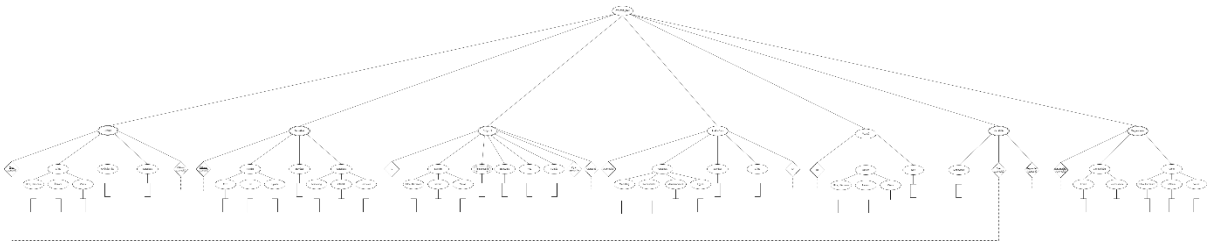
A modell 6 egyedet tartalmaz: Megrendelők, Gyárak, Dolgozók, Avalon Park, Vezető és a Termékek. Ezek között megtalálhatóak 1:1 , 1:N és N:M kapcsolatok is.

A Megrendelők és a Gyárak közötti kapcsolatnak van egy tulajdonsága(Mennyiség), amivel a rendelés méretét tudjuk kezelni.

Az ER Modell az ERDPlus használatával készült el.

1b) Az adatbázis konvertálása XDM modellre

Az XDM Modell:



A modell készítése alatt ügyeltem arra, hogy rendezett legyen és a vonalak ne keresztezzék egymást.

A PK gyémánt alakkal és aláhúzással van ábrázolva, a FK pedig szaggatott vonalas gyémánt alakkal, aláhúzással.

Amelyik egyedből több lehet, azok dupla ellipszissel lettek ábrázolva.

1c) Az XDM modell alapján XML dokumentum készítése:

Itt egyszerűen át kellett alakítani az XDM modellt XML-re.

A Gyökérelem neve C7H5VB_Hell lett.

Ahol több elem lehetett, ott mindegyikből készült 3 példány.

A példányok előtt megjegyzésként ott van a nevük átláthatóság érdekében.

Az XML kódja:

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

<C7H5VB_Hell>

<!--Gyárak-->
```

<Gyarak GyarAzonosito="1" VallalatiAzonosito="1">

<Cime>

<UtcaHazszam>Vas utca 7</UtcaHazszam>

<IrSzam>3534</IrSzam>

<Varos>Miskolc</Varos>

</Cime>

<GyartasiRata>10000</GyartasiRata>

<Kiadasok>10000000</Kiadasok>

</Gyarak>

<Gyarak GyarAzonosito="2" VallalatiAzonosito="2">

<Cime>

<UtcaHazszam>Gyár utca 1</UtcaHazszam>

<IrSzam>3525</IrSzam>

<Varos>Miskolc</Varos>

</Cime>

<GyartasiRata>15000</GyartasiRata>

<Kiadasok>13000000</Kiadasok>

</Gyarak>

<Gyarak GyarAzonosito="3" VallalatiAzonosito="3">

<Cime>

<UtcaHazszam>Hell utca 1</UtcaHazszam>

<IrSzam>2092</IrSzam>

<Varos>Budakeszi</Varos>

</Cime>

<GyartasiRata>35000</GyartasiRata>

<Kiadasok>19500000</Kiadasok>

</Gyarak>

<!--Termékek-->

<Termekek VallalatiAzonosito="1">

<Fajtaja>

<Xixo>1</Xixo>

</Fajtaja>

<Bevetel>50000000</Bevetel>

<Kiadasok>

<Marketing>1200000</Marketing>

<Otletek>1000000</Otletek>

<Teszteles>500000</Teszteles>

</Kiadasok>

</Termekek>

<Termekek VallalatiAzonosito="2">

<Fajtaja>

<Hell>1</Hell>

</Fajtaja>

<Bevetel>80000000</Bevetel>

<Kiadasok>

<Marketing>3200000</Marketing>

<Otletek>1200000</Otletek>

<Teszteles>700000</Teszteles>

</Kiadasok>

</Termekek>

<Termekek VallalatiAzonosito="3">

<Fajtaja>

<Egyeb>1</Egyeb>

</Fajtaja>

<Bevetel>7000000</Bevetel>

<Kiadasok>

<Marketing>700000</Marketing>

<Otletek>250000</Otletek>

<Teszteles>150000</Teszteles>

</Kiadasok>

</Termekek>

<!--Dolgozók-->

<Dolgozok ID="1" GyarAzonosito="1">

<Lakcim>

<UtcaHazszam>Kerek utca 7</UtcaHazszam>

<IrSzam>3533</IrSzam>

<Varos>Miskolc</Varos>

</Lakcim>

<Beosztas>Gyártósoros</Beosztas>

<Nev>Nagy Ernő</Nev>

<Fizetes>250000</Fizetes>

<Telefonszam>06307687647</Telefonszam>

<Telefonszam>06508695857</Telefonszam>

</Dolgozok>

<Dolgozok ID="2" GyarAzonosito="3">

<Lakcim>

<UtcaHazszam>Kis utca 23</UtcaHazszam>

<IrSzam>3523</IrSzam>

<Varos>Miskolc</Varos>

</Lakcim>

<Beosztas>Gyártósoros</Beosztas>

<Nev>Látó Lajos</Nev>

<Fizetes>245000</Fizetes>

<Telefonszam>06307661589</Telefonszam>

</Dolgozok>

<Dolgozok ID="3" GyarAzonosito="2">

<Lakcim>

<UtcaHazszam>Mester utca 2</UtcaHazszam>

<IrSzam>2092</IrSzam>

<Varos>Budakeszi</Varos>

</Lakcim>

<Beosztas>Takarító</Beosztas>

<Nev>Hangos Ernő</Nev>

<Fizetes>220000</Fizetes>

<Telefonszam>062096758956</Telefonszam>

</Dolgozok>

<!--AvalonPark-->

<AvalonPark Azonosito="1" VezID="1">

<Kiadasok>

<Marketing>1000000</Marketing>

<Karbantartas>2500000</Karbantartas>

<Merchandiseok>3300000</Merchandiseok>

<Egyeb>3000000</Egyeb>

</Kiadasok>

<Bevetel>60000000</Bevetel>

<Cime>Miskolc Iglói u. 15, 3519</Cime>

</AvalonPark>

<!--Vezető-->

<Vezeto VezID="1">

<Lakcim>

<UtcaHazszam>Leveles utca 12</UtcaHazszam>

<IrSzam>3520</IrSzam>

<Varos>Miskolc</Varos>

```
</Lakcim>
<Nev>Lázár Ervin</Nev>
</Vezeto>



<!--Kiszállítás-->
<Kiszallitas GyarAzonosito="1" MegrendeloAzonosito="1">
<Mennyiseg>2500</Mennyiseg>
</Kiszallitas>

<Kiszallitas GyarAzonosito="2" MegrendeloAzonosito="2">
<Mennyiseg>5000</Mennyiseg>
</Kiszallitas>

<Kiszallitas GyarAzonosito="3" MegrendeloAzonosito="3">
<Mennyiseg>1200</Mennyiseg>
</Kiszallitas>



<!--Megrendelők-->
<Megrendelok MegrendeloAzonosito="1">
<Elerhetoseg>
        <Email>ugyfelszolgalat@coop.hu</Email>
        <TelefonSzam>06307658799</TelefonSzam>
</Elerhetoseg>
<Cime>
        <UtcaHazszam>Corvin utca 7</UtcaHazszam>
        <IrSzam>3525</IrSzam>
        <Varos>Miskolc</Varos>
</Cime>
</Megrendelok>
```

<Megrendelok MegrendeloAzonosito="2">

<Elerhetoseg>

<Email>beszerzes@spar.hu</Email>

<TelefonSzam>06407658788</TelefonSzam>

</Elerhetoseg>

<Cime>

<UtcaHazszam>Magas út 26</UtcaHazszam>

<IrSzam>3533</IrSzam>

<Varos>Miskolc</Varos>

</Cime>

</Megrendelok>

<Megrendelok MegrendeloAzonosito="3">

<Elerhetoseg>

<Email>megrendelo@arena.hu</Email>

<TelefonSzam>06105758595</TelefonSzam>

</Elerhetoseg>

<Cime>

<UtcaHazszam>Kerepesi út 9</UtcaHazszam>

<IrSzam>1087</IrSzam>

<Varos>Budapest</Varos>

</Cime>

</Megrendelok>

</C7H5VB_Hell>.

1d) Az XML dokumentum alapján XMLSchema készítése - saját típusok, ref, key, keyref, speciális elemek.

Itt az XSD-ben megadjuk, hogy az XML melyik eleme milyen adat típusokat tartalmazhat, beállítjuk a kapcsolatokat FK és PK definiálásával.

Ahol az ER Modellben Composite Attribute volt, ott ComplexType segítségével hoztuk létre az element-et

Az XSD kódja:.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- Győkérelem definiálása -->
<xs:element name="C7H5VB_Hell">
<xs:complexType>
	<xs:sequence>
		<xs:element ref="Gyarak" maxOccurs="unbounded"/>
		<xs:element ref="Termekek" maxOccurs="unbounded"/>
		<xs:element ref="Dolgozok" maxOccurs="unbounded"/>
		<xs:element ref="AvalonPark" maxOccurs="1"/>
		<xs:element ref="Vezeto" maxOccurs="1"/>
		<xs:element ref="Kiszallitas" maxOccurs="unbounded"/>
		<xs:element ref="Megrendelok" maxOccurs="unbounded"/>
	</xs:sequence>
</xs:complexType>

<!-- PK és FK létrehozása -->
<xs:key name="GyarAzonosito">
	<xs:selector xpath="Gyarak"/>
	<xs:field xpath="@GyarAzonosito"/>
</xs:key>

<xs:key name="VallalatiAzonosito">
	<xs:selector xpath="Termekek"/>
	<xs:field xpath="@VallalatiAzonosito"/>
</xs:key>
<xs:keyref name="GyarakVallalatiAzonosito" refer="VallalatiAzonosito">
	<xs:selector xpath="Gyarak"/>
	<xs:field xpath="@VallalatiAzonosito"/>
</xs:keyref>

<xs:key name="ID">
	<xs:selector xpath="Dolgozok"/>
	<xs:field xpath="@ID"/>
```

```
</xs:key>
<xs:keyref name="DolgozokGyarAzonosito" refer="GyarAzonosito">
        <xs:selector xpath="Dolgozok"/>
        <xs:field xpath="@GyarAzonosito"/>
</xs:keyref>

<xs:key name="Azonosito">
        <xs:selector xpath="AvalonPark"/>
        <xs:field xpath="@Azonosito"/>
</xs:key>
<xs:keyref name="AvalonParkVezID" refer="VezID">
        <xs:selector xpath="AvalonPark"/>
        <xs:field xpath="@VezID"/>
</xs:keyref>

<xs:key name="VezID">
        <xs:selector xpath="Vezeto"/>
        <xs:field xpath="@VezID"/>
</xs:key>

<xs:keyref name="KiszallitasMegrendeloAzonosito" refer="MegrendeloAzonosito">
        <xs:selector xpath="Kiszallitas"/>
        <xs:field xpath="@MegrendeloAzonosito"/>
</xs:keyref>

<xs:keyref name="KiszallitasGyarAzonosito" refer="GyarAzonosito">
        <xs:selector xpath="Kiszallitas"/>
        <xs:field xpath="@GyarAzonosito"/>
</xs:keyref>

<xs:key name="MegrendeloAzonosito">
        <xs:selector xpath="Megrendelok"/>
        <xs:field xpath="@MegrendeloAzonosito"/>
</xs:key>

</xs:element>
```

```
<!-- C7H5VB_Hell Gyermek elemei definiálása -->
<xs:element name="Gyarak">
<xs:complexType>
	<xs:sequence>
		<xs:element name="Cime" type="GyarakCimeType"/>
		<xs:element name="GyartasiRata" type="xs:string"/>
		<xs:element name="Kiadasok" type="penzosszeg"/>
	</xs:sequence>
	<xs:attribute name="GyarAzonosito" type="xs:integer"/>
	<xs:attribute name="VallalatiAzonosito" type="xs:integer"/>
</xs:complexType>
</xs:element>

<xs:element name="Termekek">
<xs:complexType>
	<xs:sequence>
		<xs:element name="Fajtaja" type="FajtajaType"/>
		<xs:element name="Bevetel" type="penzosszeg"/>
		<xs:element name="Kiadasok" type="'TermekekKiadasokType"/>
	</xs:sequence>
	<xs:attribute name="VallalatiAzonosito" type="xs:integer"/>
</xs:complexType>
</xs:element>

<xs:element name="Dolgozok">
<xs:complexType>
	<xs:sequence>
		<xs:element name="Lakcim" type="DolgozokLakcimType"/>
		<xs:element name="Beosztas" type="xs:string"/>
		<xs:element name="Nev" type="xs:string"/>
		<xs:element name="Fizetes" type="penzosszeg"/>
		<xs:element name="'Telefonszam" type="xs:string" maxOccurs="unbounded"/>
	</xs:sequence>
	<xs:attribute name="ID" type="xs:integer"/>
```

```
        <xs:attribute name="GyarAzonosito" type="xs:integer"/>
</xs:complexType>
</xs:element>

<xs:element name="AvalonPark">
<xs:complexType>
        <xs:sequence>
                <xs:element name="Kiadasok" type="AvalonParkKiadasokType"/>
                <xs:element name="Bevetel" type="penzosszeg"/>
                <xs:element name="Cime" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="Azonosito" type="xs:integer"/>
        <xs:attribute name="VezID" type="xs:integer"/>
</xs:complexType>
</xs:element>

<xs:element name="Vezeto">
<xs:complexType>
        <xs:sequence>
                <xs:element name="Lakcim" type="DolgozokLakcimType"/>
                <xs:element name="Nev" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="VezID" type="xs:integer"/>
</xs:complexType>
</xs:element>

<xs:element name="Kiszallitas">
<xs:complexType>
        <xs:sequence>
                <xs:element name="Mennyiseg" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="GyarAzonosito" type="xs:integer"/>
        <xs:attribute name="MegrendeloAzonosito" type="xs:integer"/>
</xs:complexType>
</xs:element>
```

```
<xs:element name="Megrendelok">
<xs:complexType>
	<xs:sequence>
		<xs:element name="Elerhetoseg" type="ElerhetosegType"/>
		<xs:element name="Cime" type="DolgozokLakcimType"/>
	</xs:sequence>
	<xs:attribute name="MegrendeloAzonosito" type="xs:integer"/>
</xs:complexType>
</xs:element>


<!-- Komplex típusok definiálása -->
<xs:complexType name="GyarakCimeType">
<xs:sequence>
	<xs:element name="UtcaHazszam" type="xs:string"/>
	<xs:element name="IrSzam" type="xs:string"/>
	<xs:element name="Varos" type="xs:string"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="FajtajaType">
<xs:choice>
	<xs:element name="Xixo" type="xs:string"/>
	<xs:element name="Hell" type="xs:string"/>
	<xs:element name="Egyeb" type="xs:string"/>
</xs:choice>
</xs:complexType>

<xs:complexType name="TermekekKiadasokType">
<xs:sequence>
	<xs:element name="Marketing" type="penzosszeg"/>
	<xs:element name="Otletek" type="penzosszeg"/>
	<xs:element name="Teszteles" type="penzosszeg"/>
</xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="AvalonParkKiadasokType">
<xs:sequence>
	<xs:element name="Marketing" type="penzosszeg"/>
	<xs:element name="Karbantartas" type="penzosszeg"/>
	<xs:element name="Merchandiseok" type="penzosszeg"/>
	<xs:element name="Egyeb" type="penzosszeg"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="DolgozokLakcimType">
<xs:sequence>
	<xs:element name="UtcaHazszam" type="xs:string"/>
	<xs:element name="IrSzam" type="xs:string"/>
	<xs:element name="Varos" type="xs:string"/>
</xs:sequence>
</xs:complexType>

<xs:complexType name="ElerhetosegType">
<xs:sequence>
	<xs:element name="Email" type="xs:string"/>
	<xs:element name="TelefonSzam" type="xs:string"/>
</xs:sequence>
</xs:complexType>

<!-- Egyszerű típus definiálása -->
<xs:simpleType name="penzosszeg">
<xs:restriction base="xs:integer"/>
</xs:simpleType>

</xs:schema>
```

2. feladat

A feladat egy DOM program készítése az XML dokumentum - *XMLNeptunkod.xml* – adatainak adminisztrálása alapján:

Project name: DOMParseC7h5vb

Package: hu.domparse.c7h5vb

Class names: (DomReadC7h5vb, DomModifyC7h5vb, DomQueryC7h5vb, DOMWriteC7h5vb)

2a) adatolvasás (kód – comment együtt) – fájlnév: *DOMReadNeptunkod.java* megvalósítás

Ebben a Java DOM kódban elsőnek beolvassuk az XML fájlunkat, majd a konzolba strukturáltan kiírjuk.

A fájl beolvasása után a DocumentBuildert és a DocumentBuilderFactory használatával egy DOM fát építünk fel az XML filebol.

A kód:

```
public class DomReadC7h5vb {

   public static void main(String[] args) {

      try {

         //Beolvassa az XML fájlt

         File xmlFile = new File("XMLC7H5VB.xml");

         DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

         DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

         Document doc = dBuilder.parse(xmlFile);

         doc.getDocumentElement().normalize();


         // Gyökér elem lekérése

         Element rootElement = doc.getDocumentElement();

         System.out.println("Gyökér elem: " + rootElement.getNodeName());


         // Fa teljes tartalmának kiírása

         printNode(rootElement, "");

      } catch (Exception e) {

         e.printStackTrace();

      }

   }


   private static void printNode(Node node, String indent) {

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

```
            System.out.println(indent + "Elem: " + node.getNodeName());


            if (node.hasAttributes()) {
                NamedNodeMap attributes = node.getAttributes();
                for (int i = 0; i < attributes.getLength(); i++) {
                    Node attribute = attributes.item(i);
                    System.out.println(indent + "  Attribútum: " + attribute.getNodeName() + " = " + attribute.getNodeValue());
                }
            }


            if (node.hasChildNodes()) {
                NodeList children = node.getChildNodes();
                for (int i = 0; i < children.getLength(); i++) {
                    Node child = children.item(i);
                    printNode(child, indent + "  ");
                }
            }
        } else if (node.getNodeType() == Node.TEXT_NODE) {
            String textContent = node.getNodeValue().trim();
            if (!textContent.isEmpty()) {
                System.out.println(indent + "Tartalom: " + textContent);
            }
        }
    }
}
```

2b) adatmódosítás (kód – comment együtt) – fájlnév: *DOMModifyNeptunkod.java*

Itt miután beolvastuk az XML filet, végrehajtottunk 5 fajta módosítást:
- Dolgozóhoz telefonszám hozzáadása

- Vezető nevének módosítása

- Gyártási ráták növelése

- Eltávolítja az első megrendelőt

- Gyártósoros dolgozók fizetésének emelése

A Kód:

```
public class DomModifyC7h5vb {


    public static void main(String[] args) {
        try {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();

            // XML dokumentum beolvasása
            Document document = builder.parse("XMLC7H5VB.xml");

            // Dolgozóhoz telefonszám hozzáadása
            addPhoneNumber(document, 2, "06101234567");

            // Vezető nevének módosítása
            modifyLeaderName(document, "Nagy Ervin");

            // Gyártási ráták növelése
            increaseManufacturingRate(document, 100);

            // Eltávolítja az első megrendelőt
            removeFirstCustomer(document);

            // Gyártósoros dolgozók fizetésének emelése
            increaseSalary(document, 1500);



            // Fájlba írás és konzolra írás
            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            DOMSource source = new DOMSource(document);
```

```
        // Konzolra írás
        StreamResult console = new StreamResult(System.out);
        transformer.transform(source, console);

        // Fájlba mentés
        StreamResult file = new StreamResult("XMLC7H5VBModify.xml");
        transformer.transform(source, file);

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

private static void addPhoneNumber(Document doc, int dolgozoID, String newPhoneNumber) {
    NodeList dolgozok = doc.getElementsByTagName("Dolgozok");
    for (int i = 0; i < dolgozok.getLength(); i++) {
        Element dolgozo = (Element) dolgozok.item(i);
        if (Integer.parseInt(dolgozo.getAttribute("ID")) == dolgozoID) {
            Element phoneElem = doc.createElement("Telefonszam");
            phoneElem.setTextContent(newPhoneNumber);
            dolgozo.appendChild(phoneElem);
            break;
        }
    }
}

private static void modifyLeaderName(Document doc, String newName) {
    NodeList vezetok = doc.getElementsByTagName("Vezeto");
    if (vezetok.getLength() > 0) {
        Element vezeto = (Element) vezetok.item(0);
        vezeto.getElementsByTagName("Nev").item(0).setTextContent(newName);
```

```
        }
    }

    private static void increaseManufacturingRate(Document doc, int increaseBy) {
    NodeList gyarak = doc.getElementsByTagName("Gyarak");
        for (int i = 0; i < gyarak.getLength(); i++) {
            Element gyar = (Element) gyarak.item(i);
            String currentRate = gyar.getAttribute("GyartasiRata");
            int newRate = Integer.parseInt(currentRate) + increaseBy;
            gyar.setAttribute("GyartasiRata", Integer.toString(newRate));
        }
    }

private static void removeFirstCustomer(Document doc) {
    NodeList megrendelok = doc.getElementsByTagName("Megrendelok");
    if (megrendelok.getLength() > 0) {
        Node firstCustomer = megrendelok.item(0);
        firstCustomer.getParentNode().removeChild(firstCustomer);
    }
}

    private static void increaseSalary(Document doc, int amount) {
        NodeList dolgozok = doc.getElementsByTagName("Dolgozok");
        for (int i = 0; i < dolgozok.getLength(); i++) {
            Element dolgozo = (Element) dolgozok.item(i);
            if (dolgozo.getElementsByTagName("Beosztas").item(0).getTextContent().equals("Gyártósoros")) {
                int currentSalary = Integer.parseInt(dolgozo.getElementsByTagName("Fizetes").item(0).getTextContent());
                dolgozo.getElementsByTagName("Fizetes").item(0).setTextContent(String.valueOf(currentSalary + amount));
            }
        }
    }
}
```

2c) *adatlekérdezés* (kód – comment együtt) – fájlnév: *DOMQueryNeptunkod.java*

Itt 5 lekérdezést valósítottam meg:

- Az összes gyár azonosítójának kiírása

-Gyárak címeinek kiírása

-Összes dolgozónak a fizetésének összege

- Gyártósoros dolgozóknak a fizetésének összege

-Összetett lekérdezés: a 3000-nél nagyobb rendeléseknek a gyártójának címe és megrendelőjének a címe, valamint rendelés mérete.

A Kód:

```
package hu.domparse.c7h5vb;


import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.File;
import java.util.ArrayList;
import java.util.List;


public class DomQueryC7h5vb {
    public static void main(String[] args) {
        try {
            File xmlFile = new File("XMLC7H5VB.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);
            doc.getDocumentElement().normalize();


            System.out.println("\nGyárak:");
```

```
        List<String> gyarak = getGyarak(doc);
        for (String gyar : gyarak) {
            System.out.println(gyar);
        }

        System.out.println("\nGyárak címei:");
        List<String> cimek = getGyarCimek(doc);
        for (String cim : cimek) {
            System.out.println(cim);
        }
        System.out.println("\nDolgozók fizetéseinek összege:");
        int osszFizetes = getOsszFizetes(doc);
        System.out.println("Összesített fizetés: " + osszFizetes);

        System.out.println("\n'Gyártósoros' beosztásban dolgozók összfizetése:");
        int osszFizetesGyartosoros = getOsszFizetesGyartosoros(doc);
        System.out.println("Gyártósoros dolgozók összfizetése: " + osszFizetesGyartosoros);

        System.out.println("\nÖsszetett lekérdezés eredménye:");
        List<String> osszetettLekerdezesEredmeny = osszetettLekerdezes(doc);
        for (String eredmeny : osszetettLekerdezesEredmeny) {
            System.out.println(eredmeny);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private static List<String> getGyarak(Document doc) {
    List<String> gyarak = new ArrayList<>();
```

```
    NodeList gyarakElements = doc.getElementsByTagName("Gyarak");
    for (int i = 0; i < gyarakElements.getLength(); i++) {
        Element gyarElement = (Element) gyarakElements.item(i);
        String gyarAzonosito = gyarElement.getAttribute("GyarAzonosito");
        gyarak.add("Gyár Azonosító: " + gyarAzonosito);
    }
    return gyarak;
}

private static List<String> getGyarCimek(Document doc) {
    List<String> cimek = new ArrayList<>();
    NodeList gyarakElements = doc.getElementsByTagName("Gyarak");
    for (int i = 0; i < gyarakElements.getLength(); i++) {
        Element gyarElement = (Element) gyarakElements.item(i);
        Element cimElement = (Element) gyarElement.getElementsByTagName("Cime").item(0);
        String utca = cimElement.getElementsByTagName("UtcaHazszam").item(0).getTextContent();
        String varos = cimElement.getElementsByTagName("Varos").item(0).getTextContent();
        cimek.add("Cím: " + utca + ", " + varos);
    }
    return cimek;
}

private static List<String> osszetettLekerdezes(Document doc) {
    List<String> eredmeny = new ArrayList<>();
    NodeList kiszallitasElements = doc.getElementsByTagName("Kiszallitas");
    NodeList gyarakElements = doc.getElementsByTagName("Gyarak");
    NodeList megrendelokElements = doc.getElementsByTagName("Megrendelok");

    for (int i = 0; i < kiszallitasElements.getLength(); i++) {
        Element kiszallitasElement = (Element) kiszallitasElements.item(i);
```

```
            int mennyiseg =
Integer.parseInt(kiszallitasElement.getElementsByTagName("Mennyiseg").item(0).getTextContent());

            if (mennyiseg > 3000) {
                String gyarAzonosito = kiszallitasElement.getAttribute("GyarAzonosito");
                String megrendeloAzonosito = kiszallitasElement.getAttribute("MegrendeloAzonosito");

                String gyarCim = getCim(gyarakElements, gyarAzonosito);
                String megrendeloCim = getCim(megrendelokElements, megrendeloAzonosito);

                eredmeny.add("Mennyiség: " + mennyiseg + ", Gyár Címe: " + gyarCim + ", Megrendelő Címe: " +
megrendeloCim);
            }
        }
        return eredmeny;
    }

    private static int getOsszFizetes(Document doc) {
        int osszeg = 0;
        NodeList dolgozok = doc.getElementsByTagName("Dolgozok");
        for (int i = 0; i < dolgozok.getLength(); i++) {
            Element dolgozo = (Element) dolgozok.item(i);
            int fizetes = Integer.parseInt(dolgozo.getElementsByTagName("Fizetes").item(0).getTextContent());
            osszeg += fizetes;
        }
        return osszeg;
    }

    private static int getOsszFizetesGyartosoros(Document doc) {
        int osszeg = 0;
```

```
        NodeList dolgozok = doc.getElementsByTagName("Dolgozok");
        for (int i = 0; i < dolgozok.getLength(); i++) {
            Element dolgozo = (Element) dolgozok.item(i);
            String beosztas = dolgozo.getElementsByTagName("Beosztas").item(0).getTextContent();
            if ("Gyártósoros".equals(beosztas)) {
                int fizetes = Integer.parseInt(dolgozo.getElementsByTagName("Fizetes").item(0).getTextContent());
                osszeg += fizetes;
            }
        }
        return osszeg;
    }

    private static String getCim(NodeList elements, String azonosito) {
        for (int i = 0; i < elements.getLength(); i++) {
            Element element = (Element) elements.item(i);
            if (element.getAttribute("GyarAzonosito").equals(azonosito) ||
element.getAttribute("MegrendeloAzonosito").equals(azonosito)) {
                NodeList cimek = element.getElementsByTagName("Cime");
                if (cimek.getLength() > 0) {
                    Element cimElement = (Element) cimek.item(0);
                    String utca = cimElement.getElementsByTagName("UtcaHazszam").item(0).getTextContent();
                    String varos = cimElement.getElementsByTagName("Varos").item(0).getTextContent();
                    return utca + ", " + varos;
                }
            }
        }
        return "Cím nem található";
    }

}
```

.

2d) *adatírás* - készítsen egy DOM API programot, amely egy *XMLNeptunkod.xml* dokumentum tartalmát fa struktúra formában kiírja a *konzolra és egy XMLNeptunkod1.xml* fájlba. (kód – comment együtt) – fájlnév: DOMWriteNeptunkod.java
Miután az XML fájlt beolvastuk TransformerFactory, DOMSource és StreamResult segítségével mentjük fájlba, majd kiírjuk strukturáltan a konzolba.

A Kód:

```
package hu.domparse.c7h5vb;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
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;

public class DomWriteC7h5vb {

   public static void main(String[] args) {
      try {
         DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
         DocumentBuilder builder = factory.newDocumentBuilder();

         // XML dokumentum beolvasása
         Document document = builder.parse(new File("XMLC7H5VB.xml"));
         document.getDocumentElement().normalize();

         // Mentés Új file-ba
         TransformerFactory transformerFactory = TransformerFactory.newInstance();
         Transformer transformer = transformerFactory.newTransformer();
         DOMSource source = new DOMSource(document);
         StreamResult fileResult = new StreamResult(new File("XMLC7H5VB1.xml"));
         transformer.transform(source, fileResult);

         // Konzolra írás
         StreamResult console = new StreamResult(System.out);
```

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

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

}
```

Dr. Bednarik László

tárgyjegyző