JEGYZŐKÖNYV

Webes adatkezelő környezetek

Féléves feladat

Vállalat

Készítette: **Ilyó-Kovács Levente**

Neptunkód: **NLFUA8**

Dátum: 2024.12.02

Miskolc, 2024

Bevezetés

Ez a beadandó az autókölcsönzési rendszerek XML alapú modellezését mutatja be. A rendszer különféle entitásokat ábrázol, úgymint a gépjárművek, vásárlók, kölcsönző cégek, szerelők, valamint a karbantartási és kölcsönzési események, továbbá megvalósítja ezek kapcsolatainak modelljét. Az XML használata biztosítja, hogy az adatok jól strukturált, platformfüggetlen módon tárolódjanak, amely elősegíti a rendszer egyszerű bővíthetőségét és a kapcsolatok könnyű kezelhetőségét. A modell célja, hogy a gépjárműpark, a hozzá kapcsolódó egyedi nyilvántartási adatok, a karbantartási tevékenységek, a kiadások és a bérlések átláthatóan jelenjenek meg, így lehetővé téve az adatok hatékony feldolgozását, elemzését és a szolgáltatások optimalizálását.

Feladat leírása

A féléves beadandó témája egy autókölcsönző vállalati rendszer adatmodelljének létrehozása és XML formátumban történő megvalósítása. A projekt olyan szervezeti egységeket és attribútumokat modellerez, mint a gépjárművek (márkával, típussal, évjárattal és egyéb paraméterekkel), az egyedi nyilvántartás, a vásárlók (akik a járműveket bérlik), a szerelők (akik a karbantartásokat végzik), valamint a kölcsönző cégek (melyek a gépjárműveket kölcsönadják). Az adatmodellben a kölcsönzés, a karbantartás és a kiadások eseményei logikus kapcsolatban állnak ezen entitásokkal. Az XML lehetővé teszi az adatok könnyen áttekinthető, kereshető és módosítható formában való tárolását, így biztosítva a rendszerben kezelt adatok integritását, rugalmasságát és átláthatóságát. Az így elkészült dokumentumból könnyedén lehet kiolvasni, új adatokat hozzáadni, illetve a meglévőket módosítani, támogatva az üzleti döntéshozatalt és a folyamatos fejlesztést.

1. Adatbázis ER modell készítése

Az ERDPlus online modellező eszközt vettem igénybe az ER modell elkészítéséhez, amely egyszerűvé és gyorssá teszi az entitások, attribútumok, valamint kapcsolatok ábrázolását. Az ERDPlus intuitív felülete lehetővé tette számomra, hogy hatékonyan megalkossam a vállalati szervezeti struktúrát bemutató diagramot, amely világosan szemlélteti az elemek közötti összefüggéseket.

1.1 Az egyedek tulajdonságai és az ER modell

Az alábbi felsorolás az autós kölcsönző rendszer entitásait és azok tulajdonságait mutatja be az ER modell alapján. Minden egyedhez megadjuk az elsődleges kulcsot és a releváns tulajdonságokat, illetve az összetett mezőket.

Gépjármű (Gepjarmu)

- JarmuID: A Gépjármű egyed elsődleges kulcsa
- Rendszam: A gépjármű egyedi azonosító rendszáma
- Marka: A gépjármű márkája (pl. BMW, Audi)
- Evjarat: A gépjármű gyártási éve (gYear)
- GyartasiHely: A gépjármű gyártási helye (ország, város)
- Tipus: A gépjármű típusa (pl. Sedan, SUV)
- MegtettKilometer: A gépjármű által megtett összes kilométer

Egyedi nyilvántartás (EgyediNyilvantartas)

- JarmuID: Kapcsolat a Gépjárműhöz (idegen kulcs)
- Rendszam: A gépjármű rendszáma
- KGFB: Kötelező gépjármű-felelősségbiztosítás azonosítója
- Tulajdonos: A gépjármű tulajdonosának neve

Vásárló (Vasarlo)

- VasarloID: A Vásárló egyed elsődleges kulcsa
- Nev: A vásárló neve
- Lakcim: A vásárló címe (egyszerű string)
- Telefonszam: A vásárló telefonszáma

Kölcsönzés (Kolcsonzes)

- GepjarmuID: Kapcsolat a Gépjárműhöz (idegen kulcs)
- VasarloID: Kapcsolat a Vásárlóhoz (idegen kulcs)
- Tol: Kölcsönzés kezdő dátuma
- Ig: Kölcsönzés vége dátuma
- ElteltNapok: A kölcsönzéssel eltelt napok száma (0-365 között)
- KolcsonzesiAr: A kölcsönzés díja (decimális érték)

Szerelő (Szerelo)

- SzereloID: A Szerelő egyed elsődleges kulcsa
- Nev: A szerelő neve
- Lakcim: A szerelő lakcíme

- Telefonszam: A szerelő telefonszáma

Kölcsönző cég (KolcsonoCeg)

- CegID: A Kölcsönző cég egyed elsődleges kulcsa
- Cim: A cég címe
- Alapterulet: Az iroda alapterülete
- AutokSzama: A cég flottájában lévő autók száma

Karbantartás (Karbantartas)

- GepjarmuID: Kapcsolat a Gépjárműhöz (idegen kulcs)
- SzereloID: Kapcsolat a Szerelőhöz (idegen kulcs)
- Datum: A karbantartás dátuma
- MegtettKilometer: A karbantartás idején rögzített kilométeróra állás

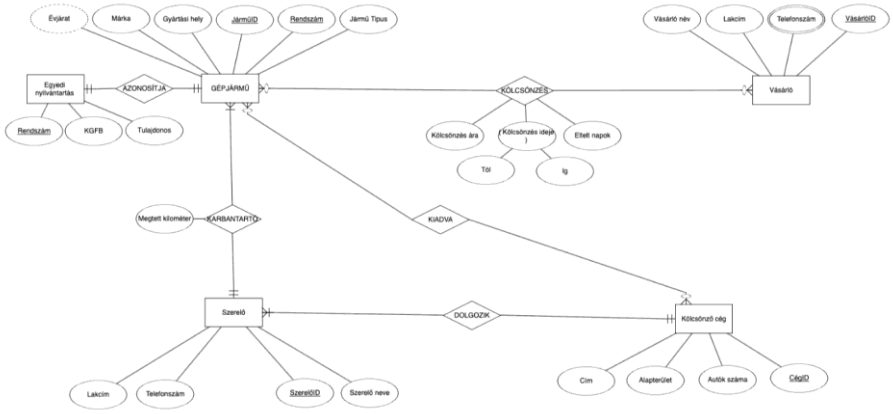
Dolgozik (Dolgozik)

- SzereloID: Kapcsolat a Szerelőhöz (idegen kulcs)
- CegID: Kapcsolat a Kölcsönző céghez (idegen kulcs)
- KezdoDatum: A szerelő munkaviszonyának kezdő dátuma a cégnél

Kiadva (Kiadva)

- GepjarmuID: Kapcsolat a Gépjárműhöz (idegen kulcs)
- CegID: Kapcsolat a Kölcsönző céghez (idegen kulcs)
- Datum: Az a dátum, amikor a gépjármű adott céghez került (pl. átadás, szerződéskötés időpontja)

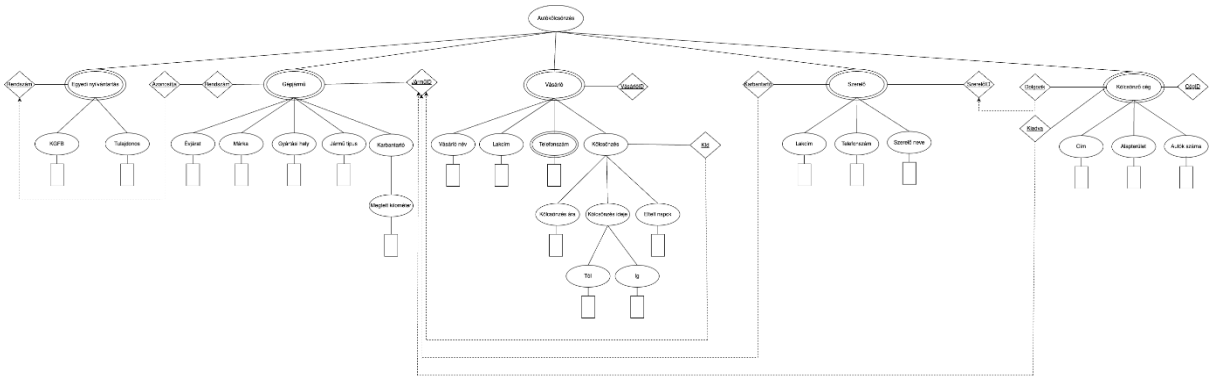
Az így definiált entitások és tulajdonságok alkotják az autós kölcsönző rendszer ER modelljét, amely később XML sémában (XSD) és XML dokumentumban kerül megjelenítésre.



1.2 Az adatbázis konvertálása XDM modellre

Az XDM modellben különféle jelöléseket használunk: ellipszist, rombuszt és téglalapot. Az ellipszis az elemeket szimbolizálja, amelyek egyedekből vagy tulajdonságokból jönnek létre. A rombusz a kulcstulajdonságokból létrejövő attribútumokat ábrázolja. A téglalap a szövegeket képviseli, amelyek később az XML dokumentumban kerülnek felhasználásra. Az ismétlődő elemeket dupla ellipszissel jelöljük. A kulcsok és az idegen kulcsok közötti kapcsolatok szaggatott nyílvonalakkal vannak megjelenítve.

XDM modell:



1.3 Az XDM modell alapján XML dokumentum készítése

Az **XML** dokumentumot az **XDM** modell alapján hoztam létre. Először a fő elemet definiáltam, amelyet „**AutokolcsonzesRendszer**” néven neveztem el. Ezt követően a gyermekelemeket különböző megközelítésekkel alakítottam át példányokká.

```
<?xml version="1.0" encoding="UTF-8"?>
<AutokolcsonzesRendszer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="autokolcsonzes.xsd">

  <!-- 3 Gépjármű példány -->
  <Gepjarmu JarmuID="CAR1">
    <Rendszam>ABC100</Rendszam>
    <Marka>BMW</Marka>
    <Evjarat>2020</Evjarat>
    <GyartasiHely>Germany</GyartasiHely>
    <Tipus>Sedan</Tipus>
    <MegtettKilometer>10000</MegtettKilometer>
  </Gepjarmu>

  <Gepjarmu JarmuID="CAR2">
    <Rendszam>ABC200</Rendszam>
    <Marka>Audi</Marka>
    <Evjarat>2021</Evjarat>
    <GyartasiHely>Hungary</GyartasiHely>
    <Tipus>SUV</Tipus>
    <MegtettKilometer>20000</MegtettKilometer>
  </Gepjarmu>

  <Gepjarmu JarmuID="CAR3">
    <Rendszam>ABC300</Rendszam>
    <Marka>Toyota</Marka>
```

```xml
  <Evjarat>2019</Evjarat>
  <GyartasiHely>Japan</GyartasiHely>
  <Tipus>Hatchback</Tipus>
  <MegtettKilometer>50000</MegtettKilometer>
</Gepjarmu>

<!-- 3 Egyedi nyilvántartás példány -->
<EgyediNyilvantartas JarmuID="CAR1">
  <Rendszam>ABC100</Rendszam>
  <KGFB>KGFB12345</KGFB>
  <Tulajdonos>Kis János</Tulajdonos>
</EgyediNyilvantartas>

<EgyediNyilvantartas JarmuID="CAR2">
  <Rendszam>ABC200</Rendszam>
  <KGFB>KGFB54321</KGFB>
  <Tulajdonos>Balogh Éva</Tulajdonos>
</EgyediNyilvantartas>

<EgyediNyilvantartas JarmuID="CAR3">
  <Rendszam>ABC300</Rendszam>
  <KGFB>KGFB67890</KGFB>
  <Tulajdonos>Farkas Péter</Tulajdonos>
</EgyediNyilvantartas>

<!-- 3 Vásárló példány -->
<Vasarlo VasarloID="V1">
  <Nev>Kovács István</Nev>
  <Lakcim>Budapest</Lakcim>
  <Telefonszam>1234567</Telefonszam>
</Vasarlo>

<Vasarlo VasarloID="V2">
  <Nev>Nagy Péter</Nev>
  <Lakcim>Debrecen</Lakcim>
  <Telefonszam>2234567</Telefonszam>
</Vasarlo>

<Vasarlo VasarloID="V3">
  <Nev>Tóth Anna</Nev>
  <Lakcim>Szeged</Lakcim>
  <Telefonszam>3234567</Telefonszam>
</Vasarlo>

<!-- 3 Kölcsönzés példány -->
<Kolcsonzes GepjarmuID="CAR1" VasarloID="V1">
  <Tol>2021-06-01</Tol>
  <Ig>2021-06-10</Ig>
  <ElteltNapok>9</ElteltNapok>
  <KolcsonzesiAr>200.50</KolcsonzesiAr>
</Kolcsonzes>

<Kolcsonzes GepjarmuID="CAR2" VasarloID="V2">
  <Tol>2021-07-05</Tol>
  <Ig>2021-07-09</Ig>
  <ElteltNapok>4</ElteltNapok>
  <KolcsonzesiAr>150.00</KolcsonzesiAr>
</Kolcsonzes>

<Kolcsonzes GepjarmuID="CAR3" VasarloID="V3">
  <Tol>2021-08-15</Tol>
  <Ig>2021-08-20</Ig>
  <ElteltNapok>5</ElteltNapok>
```

```
  <KolcsonzesiAr>180.75</KolcsonzesiAr>
</Kolcsonzes>

<!-- 3 Szerelő példány -->
<Szerelo SzereloID="SZ1">
  <Nev>Horváth Béla</Nev>
  <Lakcim>Debrecen</Lakcim>
  <Telefonszam>2345678</Telefonszam>
</Szerelo>

<Szerelo SzereloID="SZ2">
  <Nev>Szabó Katalin</Nev>
  <Lakcim>Szeged</Lakcim>
  <Telefonszam>3456789</Telefonszam>
</Szerelo>

<Szerelo SzereloID="SZ3">
  <Nev>Mészáros László</Nev>
  <Lakcim>Pécs</Lakcim>
  <Telefonszam>4567890</Telefonszam>
</Szerelo>

<!-- 3 Kölcsönző cég példány -->
<KolcsonoCeg CegID="CEG1">
  <Cim>Budapest, XY utca 10</Cim>
  <Alapterulet>250.5</Alapterulet>
  <AutokSzama>10</AutokSzama>
</KolcsonoCeg>

<KolcsonoCeg CegID="CEG2">
  <Cim>Győr, AB tér 5</Cim>
  <Alapterulet>300.0</Alapterulet>
  <AutokSzama>20</AutokSzama>
</KolcsonoCeg>

<KolcsonoCeg CegID="CEG3">
  <Cim>Miskolc, CD u. 7</Cim>
  <Alapterulet>150.75</Alapterulet>
  <AutokSzama>15</AutokSzama>
</KolcsonoCeg>

<!-- 3 Karbantartás példány -->
<Karbantartas GepjarmuID="CAR1" SzereloID="SZ1">
  <Datum>2022-01-15</Datum>
  <MegtettKilometer>15000</MegtettKilometer>
</Karbantartas>

<Karbantartas GepjarmuID="CAR2" SzereloID="SZ2">
  <Datum>2022-03-20</Datum>
  <MegtettKilometer>25000</MegtettKilometer>
</Karbantartas>

<Karbantartas GepjarmuID="CAR3" SzereloID="SZ3">
  <Datum>2023-07-10</Datum>
  <MegtettKilometer>60000</MegtettKilometer>
</Karbantartas>

<!-- 3 Dolgozik példány -->
<Dolgozik SzereloID="SZ1" CegID="CEG1">
  <KezdoDatum>2020-01-01</KezdoDatum>
</Dolgozik>

<Dolgozik SzereloID="SZ2" CegID="CEG2">
```

```
    <KezdoDatum>2021-02-01</KezdoDatum>
  </Dolgozik>

  <Dolgozik SzereloID="SZ3" CegID="CEG3">
    <KezdoDatum>2021-05-10</KezdoDatum>
  </Dolgozik>

  <!-- 3 Kiadva példány -->
  <Kiadva GepjarmuID="CAR1" CegID="CEG1">
    <Datum>2021-03-15</Datum>
  </Kiadva>

  <Kiadva GepjarmuID="CAR2" CegID="CEG2">
    <Datum>2021-08-20</Datum>
  </Kiadva>

  <Kiadva GepjarmuID="CAR3" CegID="CEG3">
    <Datum>2022-11-05</Datum>
  </Kiadva>

</AutokolcsonzesRendszer>
```

1.4 Az XML dokumentum alapján XMLSchema készítése

Az autókölcsönző rendszert leíró XML dokumentumhoz készítettem egy, az adatok érvényességét biztosító XML séma definíciót (XSD). Először meghatároztam az egyszerű típusokat, amelyek az egyes adatmezők lehetséges értékeit és formátumát szabályozzák. Például a rendszám típusnál reguláris kifejezéssel határoztam meg a járművek rendszámának helyes formátumát. A napok számát korlátozó típusok esetében minimális és maximális határértékeket adtam meg, míg a járműmárkákra enumerációt alkalmaztam, hogy csak előre meghatározott értékek lehessenek.

Ezután kialakítottam a komplex típusokat minden főbb elemre, mint például a Gépjármű, EgyediNyilvantartas, Vásárló, Kölcsönzés, Karbantartas, KolcsonoCeg, valamint a Szerelo és a Dolgozik, Kiadva elemekre. Mindezek a típusok al-elemekből, attribútumokból és különféle hivatkozásokból állnak, és együtt jelenítik meg az autókölcsönző üzleti struktúráját. A sémán belül megadtam az egyedi azonosító attribútumokat is (például a JarmuID, VasarloID, SzereloID, CegID), amelyek segítségével minden entitás egyértelműen azonosítható.

A séma elsődleges és idegen kulcs jellegű hivatkozásokat is tartalmaz. Például a Kölcsönzés vagy az EgyediNyilvantartas elemeknél a JarmuID segítségével kötöttem össze a Gépjármű rekordokat a megfelelő adatokkal, így biztosítva az integritást és a konzisztens adatbázis-szerkezetet. Hasonló módon a Kiadva vagy Dolgozik elemek is IDREF attribútumokon keresztül kapcsolódnak a megfelelő cégekhez és szerelőkhöz. Ennek köszönhetően a rendszerben minden járműhöz, karbantartáshoz, kölcsönzéshez és egyéb művelethez pontosan megadható, mely entitásokhoz tartozik, és ezáltal garantált az adatok helyessége és következetessége.

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

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

	<!-- Napok korlátozása min-max értékkel -->
	<xs:simpleType name="NapokType">
		<xs:restriction base="xs:integer">
			<xs:minExclusive value="0" />
			<xs:maxExclusive value="365" />
		</xs:restriction>
	</xs:simpleType>

	<!-- Márka enumeráció -->
	<xs:simpleType name="MarkaType">
		<xs:restriction base="xs:string">
			<xs:enumeration value="BMW" />
			<xs:enumeration value="Audi" />
			<xs:enumeration value="Mercedes" />
			<xs:enumeration value="Toyota" />
			<xs:enumeration value="Ford" />
		</xs:restriction>
	</xs:simpleType>

	<!-- Gépjármű -->
	<xs:complexType name="GepjarmuType">
		<xs:sequence>
			<xs:element name="Rendszam" type="RendszamType" />
			<xs:element name="Marka" type="MarkaType" />
			<xs:element name="Evjarat" type="xs:gYear" />
			<xs:element name="GyartasiHely" type="xs:string" />
			<xs:element name="Tipus" type="xs:string" />
			<xs:element name="MegtettKilometer" type="xs:integer" />
		</xs:sequence>
		<xs:attribute name="JarmuID" type="xs:ID" use="required" />
	</xs:complexType>

	<!-- Egyedi nyilvántartás -->
	<xs:complexType name="EgyediNyilvantartasType">
		<xs:sequence>
			<xs:element name="Rendszam" type="RendszamType" />
			<xs:element name="KGFB" type="xs:string" />
			<xs:element name="Tulajdonos" type="xs:string" />
		</xs:sequence>
		<!-- Azonosítja kapcsolat: a JarmuID alapján kötjük össze -->
		<xs:attribute name="JarmuID" type="xs:IDREF" use="required"/>
	</xs:complexType>

	<!-- Vásárló -->
	<xs:complexType name="VasarloType">
		<xs:sequence>
			<xs:element name="Nev" type="xs:string" />
			<xs:element name="Lakcim" type="xs:string" />
			<xs:element name="Telefonszam" type="xs:string" />
		</xs:sequence>
		<xs:attribute name="VasarloID" type="xs:ID" use="required" />
	</xs:complexType>
```

```
<!-- Kölcsönzés -->
<xs:complexType name="KolcsonzesType">
	<xs:sequence>
		<xs:element name="Tol" type="xs:date" />
		<xs:element name="Ig" type="xs:date" />
		<xs:element name="ElteltNapok" type="NapokType" />
		<xs:element name="KolcsonzesiAr" type="xs:decimal" />
	</xs:sequence>
	<xs:attribute name="GepjarmuID" type="xs:IDREF" use="required" />
	<xs:attribute name="VasarloID" type="xs:IDREF" use="required" />
</xs:complexType>

<!-- Szerelő -->
<xs:complexType name="SzereloType">
	<xs:sequence>
		<xs:element name="Nev" type="xs:string" />
		<xs:element name="Lakcim" type="xs:string" />
		<xs:element name="Telefonszam" type="xs:string" />
	</xs:sequence>
	<xs:attribute name="SzereloID" type="xs:ID" use="required" />
</xs:complexType>

<!-- Kölcsönző cég -->
<xs:complexType name="KolcsonoCegType">
	<xs:sequence>
		<xs:element name="Cim" type="xs:string" />
		<xs:element name="Alapterulet" type="xs:decimal" />
		<xs:element name="AutokSzama" type="xs:integer" />
	</xs:sequence>
	<xs:attribute name="CegID" type="xs:ID" use="required" />
</xs:complexType>

<!-- Karbantartás -->
<xs:complexType name="KarbantartasType">
	<xs:sequence>
		<xs:element name="Datum" type="xs:date" />
		<xs:element name="MegtettKilometer" type="xs:integer" />
	</xs:sequence>
	<xs:attribute name="GepjarmuID" type="xs:IDREF" use="required" />
	<xs:attribute name="SzereloID" type="xs:IDREF" use="required" />
</xs:complexType>

<!-- Dolgozik -->
<xs:complexType name="DolgozikType">
	<xs:sequence>
		<xs:element name="KezdoDatum" type="xs:date" />
	</xs:sequence>
	<xs:attribute name="SzereloID" type="xs:IDREF" use="required" />
	<xs:attribute name="CegID" type="xs:IDREF" use="required" />
</xs:complexType>

<!-- Kiadva -->
<xs:complexType name="KiadvaType">
	<xs:sequence>
		<xs:element name="Datum" type="xs:date" />
	</xs:sequence>
	<xs:attribute name="GepjarmuID" type="xs:IDREF" use="required" />
	<xs:attribute name="CegID" type="xs:IDREF" use="required" />
</xs:complexType>

<!-- Gyökér elem -->
<xs:element name="AutokolcsonzesRendszer">
	<xs:complexType>
```

```
                <xs:sequence>
                        <xs:element name="Gepjarmu" type="GepjarmuType"
maxOccurs="unbounded" />
                        <xs:element name="EgyediNyilvantartas"
type="EgyediNyilvantartasType" maxOccurs="unbounded" />
                        <xs:element name="Vasarlo" type="VasarloType"
maxOccurs="unbounded" />
                        <xs:element name="Kolcsonzes" type="KolcsonzesType"
maxOccurs="unbounded" />
                        <xs:element name="Szerelo" type="SzereloType"
maxOccurs="unbounded" />
                        <xs:element name="KolcsonoCeg" type="KolcsonoCegType"
maxOccurs="unbounded" />
                        <xs:element name="Karbantartas" type="KarbantartasType"
maxOccurs="unbounded" />
                        <xs:element name="Dolgozik" type="DolgozikType"
maxOccurs="unbounded" />
                        <xs:element name="Kiadva" type="KiadvaType"
maxOccurs="unbounded" />
                </xs:sequence>
        </xs:complexType>

        <!-- Kulcsok -->
        <xs:key name="GepjarmuKey">
                <xs:selector xpath="Gepjarmu" />
                <xs:field xpath="@JarmuID" />
        </xs:key>
        <xs:key name="VasarloKey">
                <xs:selector xpath="Vasarlo" />
                <xs:field xpath="@VasarloID" />
        </xs:key>
        <xs:key name="SzereloKey">
                <xs:selector xpath="Szerelo" />
                <xs:field xpath="@SzereloID" />
        </xs:key>
        <xs:key name="CegKey">
                <xs:selector xpath="KolcsonoCeg" />
                <xs:field xpath="@CegID" />
        </xs:key>

        <!-- Kulcs referenciák -->
        <xs:keyref name="EgyediNyilvantartasGepjarmuRef"
refer="GepjarmuKey">
                <xs:selector xpath="EgyediNyilvantartas"/>
                <xs:field xpath="@JarmuID"/>
        </xs:keyref>

        <xs:keyref name="KolcsonzesGepjarmuRef" refer="GepjarmuKey">
                <xs:selector xpath="Kolcsonzes" />
                <xs:field xpath="@GepjarmuID" />
        </xs:keyref>
        <xs:keyref name="KolcsonzesVasarloRef" refer="VasarloKey">
                <xs:selector xpath="Kolcsonzes" />
                <xs:field xpath="@VasarloID" />
        </xs:keyref>
        <xs:keyref name="KarbantartasGepjarmuRef" refer="GepjarmuKey">
                <xs:selector xpath="Karbantartas" />
                <xs:field xpath="@GepjarmuID" />
        </xs:keyref>
        <xs:keyref name="KarbantartasSzereloRef" refer="SzereloKey">
                <xs:selector xpath="Karbantartas" />
                <xs:field xpath="@SzereloID" />
        </xs:keyref>
```

```
            <xs:keyref name="DolgozikSzereloRef" refer="SzereloKey">
                  <xs:selector xpath="Dolgozik" />
                  <xs:field xpath="@SzereloID" />
            </xs:keyref>
            <xs:keyref name="DolgozikCegRef" refer="CegKey">
                  <xs:selector xpath="Dolgozik" />
                  <xs:field xpath="@CegID" />
            </xs:keyref>
            <xs:keyref name="KiadvaGepjarmuRef" refer="GepjarmuKey">
                  <xs:selector xpath="Kiadva" />
                  <xs:field xpath="@GepjarmuID" />
            </xs:keyref>
            <xs:keyref name="KiadvaCegRef" refer="CegKey">
                  <xs:selector xpath="Kiadva" />
                  <xs:field xpath="@CegID" />
            </xs:keyref>
      </xs:element>
</xs:schema>
```

2. DOM program készítése JAVA környezetbe

A feladatkiírásnak megfelelően a **DOM** programokat **Java** nyelven valósítottam meg. Az elkészített programokat az alábbi alfejezetekben részletesen bemutatom.

2.1 DOM Adatolvasás

DomReadNLFUA8:

A kód egy Java alkalmazás, amely XML dokumentumot dolgoz fel a **DOM (Document Object Model)** segítségével. A cél az XML adatok elemzése, kiírása, és opcionálisan egy új fájlba való mentése. A kód egy **autókölcsönzési rendszer** adatait jeleníti meg különféle entitások (például járművek, vásárlók, szerelők) szerint.

1. **Főprogram (main metódus):**
    - A File osztállyal megadja az XML fájl elérési útját: src/resources/XMLNLFUA8.xml.
    - A buildDocument függvény segítségével betölti és normalizálja az XML dokumentumot egy Document objektumba.
    - A gyökérelem (Document.getDocumentElement().getNodeName()) nevét kiírja a konzolra.
    - Egyesével végigmegy az XML fájl különböző elemein, mint például:
        - **Gépjármű (Gepjarmu)**
        - **Egyedi Nyilvántartás (EgyediNyilvantartas)**
        - **Vásárló (Vasarlo)**
        - **Kölcsönzés (Kolcsonzes)**

- **Szerelő (Szerelo)**
- **Kölcsönző Cég (KolcsonoCeg)**
- **Karbantartás (Karbantartas)**
- **Dolgozik (Dolgozik)**
- **Kiadva (Kiadva)**

Minden elemhez egy külön függvényt hív meg, amely kinyeri és kiírja az adott típushoz tartozó adatokat.

2. **Segédfüggvények:**
    - **buildDocument:** Felépíti a DOM struktúrát az XML fájlból.
    - **printToFile:** Az aktuális DOM fájlt visszaírja egy új XML fájlba.
    - **getTextContent:** Egy adott szülőelemből (Element) lekérdezi egy gyerekelem (tagName) szöveges tartalmát.
3. **Speciális entitások feldolgozása:** Minden egyes entitásnak (pl. **Gépjármű, Vásárló**) van saját kiírófüggvénye, amely:
    - Lekéri az attribútumokat, például JarmuID, VasarloID.
    - Lekérdezi az egyes gyerekelemek szöveges tartalmát, például Rendszam, Nev.
    - Kiírja a lekért adatokat formázott szövegként a konzolra.
4. **Kiemelt DOM műveletek:**
    - **getElementsByTagName:** Az adott típusú elemek összegyűjtése (pl. Gepjarmu elemek listája).
    - **Node.getNodeType:** Ellenőrzés, hogy az aktuális csomópont ELEMENT_NODE típusú-e.
    - **Element.getAttribute:** Attribútumok kiolvasása.
    - **Node.getTextContent:** Egy adott gyerekelem szövegtartalmának lekérése.

Fontos megfigyelések

1. **Normalizálás:** A doc.getDocumentElement().normalize() hívás biztosítja, hogy az XML dokumentum feldolgozása konzisztens legyen. Az egyes szöveges csomópontokat (pl. whitespace) egyesíti.
2. **Kód strukturáltsága:**
    - Az entitás-specifikus függvények megkönnyítik az XML feldolgozás logikájának áttekinthetőségét.

- A függvények használata minimalizálja a duplikált kódot, hiszen az egyes attribútum- vagy gyerekelem-lekérések könnyen módosíthatók.

3. **Hibatűrés:** A try-catch blokk kezeli az esetleges hibákat (például fájl elérési probléma, XML feldolgozási hiba).

Fontosnak tartott kódrészlet:

kiirGepjarmuInfo függvény

Miért fontos?

Ez a függvény jól reprezentálja az entitások feldolgozásának logikáját:

- Lekérdezi az **attribútumokat** (JarmuID).
- Lekérdezi a **gyerekelemek szövegtartalmát** (pl. Rendszam, Marka).
- Formázottan kiírja az adatokat. Ez a minta egyszerűsíti más entitások hasonló feldolgozását.

```
package hu.domparse.nlfua8;

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

import org.w3c.dom.NodeList;

public class DomReadNLFUA8 {
    public static void main(String[] args) {
        try {
            // Az XML fájl elérési útjának megadása
            File xmlFile = new
File("src/resources/XMLNLFUA8.xml");

            // Az XML dokumentum felépítése (DOM objektum
létrehozása)
            Document doc = buildDocument(xmlFile);

            // A gyökér elem kiírása
```

```
            System.out.println("Gyökér elem: " +
doc.getDocumentElement().getNodeName());

            // Gepjarmu elemek beolvasása és kiírása
            System.out.println("\n<!-- Gépjármű példányok --
>");
            NodeList gepjarmuList =
doc.getElementsByTagName("Gepjarmu");
            for (int i = 0; i < gepjarmuList.getLength(); i++)
{
                Node gepjarmuNode = gepjarmuList.item(i);
                if (gepjarmuNode.getNodeType() ==
Node.ELEMENT_NODE) {
                    Element gepjarmu = (Element) gepjarmuNode;
                    kiirGepjarmuInfo(gepjarmu);
                }
            }

            // EgyediNyilvantartas elemek beolvasása és
kiírása
            System.out.println("\n<!-- Egyedi Nyilvántartás
példányok -->");
            NodeList egyediList =
doc.getElementsByTagName("EgyediNyilvantartas");
            for (int i = 0; i < egyediList.getLength(); i++) {
                Node egyediNode = egyediList.item(i);
                if (egyediNode.getNodeType() ==
Node.ELEMENT_NODE) {
                    Element egyedi = (Element) egyediNode;
                    kiirEgyediNyilvantartasInfo(egyedi);
                }
            }

            // Vasarlo elemek beolvasása és kiírása
            System.out.println("\n<!-- Vásárló példányok --
>");
            NodeList vasarloList =
doc.getElementsByTagName("Vasarlo");
            for (int i = 0; i < vasarloList.getLength(); i++)
{
                Node vasarloNode = vasarloList.item(i);
                if (vasarloNode.getNodeType() ==
Node.ELEMENT_NODE) {
                    Element vasarlo = (Element) vasarloNode;
                    kiirVasarloInfo(vasarlo);
                }
```

```
            }

            // Kolcsonzes elemek beolvasása és kiírása
            System.out.println("\n<!-- Kölcsönzés példányok --
>");
            NodeList kolcsonzesList =
doc.getElementsByTagName("Kolcsonzes");
            for (int i = 0; i < kolcsonzesList.getLength();
i++) {
                Node kolcsonzesNode = kolcsonzesList.item(i);
                if (kolcsonzesNode.getNodeType() ==
Node.ELEMENT_NODE) {
                    Element kolcsonzes = (Element)
kolcsonzesNode;
                    kiirKolcsonzesInfo(kolcsonzes);
                }
            }

            // Szerelo elemek beolvasása és kiírása
            System.out.println("\n<!-- Szerelő példányok --
>");
            NodeList szereloList =
doc.getElementsByTagName("Szerelo");
            for (int i = 0; i < szereloList.getLength(); i++)
{
                Node szereloNode = szereloList.item(i);
                if (szereloNode.getNodeType() ==
Node.ELEMENT_NODE) {
                    Element szerelo = (Element) szereloNode;
                    kiirSzereloInfo(szerelo);
                }
            }

            // KolcsonoCeg elemek beolvasása és kiírása
            System.out.println("\n<!-- Kölcsönző Cég példányok
-->");
            NodeList cegList =
doc.getElementsByTagName("KolcsonoCeg");
            for (int i = 0; i < cegList.getLength(); i++) {
                Node cegNode = cegList.item(i);
                if (cegNode.getNodeType() ==
Node.ELEMENT_NODE) {
                    Element ceg = (Element) cegNode;
                    kiirCegInfo(ceg);
                }
            }
```

```
            // Karbantartas elemek beolvasása és kiírása
            System.out.println("\n<!-- Karbantartás példányok
-->");
            NodeList karbList =
doc.getElementsByTagName("Karbantartas");
            for (int i = 0; i < karbList.getLength(); i++) {
                Node karbNode = karbList.item(i);
                if (karbNode.getNodeType() ==
Node.ELEMENT_NODE) {
                    Element karb = (Element) karbNode;
                    kiirKarbantartasInfo(karb);
                }
            }

            // Dolgozik elemek beolvasása és kiírása
            System.out.println("\n<!-- Dolgozik példányok --
>");
            NodeList dolgozikList =
doc.getElementsByTagName("Dolgozik");
            for (int i = 0; i < dolgozikList.getLength(); i++)
{
                Node dolgozikNode = dolgozikList.item(i);
                if (dolgozikNode.getNodeType() ==
Node.ELEMENT_NODE) {
                    Element dolgozik = (Element) dolgozikNode;
                    kiirDolgozikInfo(dolgozik);
                }
            }

            // Kiadva elemek beolvasása és kiírása
            System.out.println("\n<!-- Kiadva példányok -->");
            NodeList kiadvaList =
doc.getElementsByTagName("Kiadva");
            for (int i = 0; i < kiadvaList.getLength(); i++) {
                Node kiadvaNode = kiadvaList.item(i);
                if (kiadvaNode.getNodeType() ==
Node.ELEMENT_NODE) {
                    Element kiadva = (Element) kiadvaNode;
                    kiirKiadvaInfo(kiadva);
                }
            }

            // Fájlba írás (opcionális)
            printToFile(doc);
```

```
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    private static void printToFile(Document doc) throws
Exception {
        File outputFile = new
File("src/resources/AutokolcsonzesRendszer_output.xml");
        Transformer transformer =
TransformerFactory.newInstance().newTransformer();
        DOMSource source = new DOMSource(doc);
        StreamResult outFile = new StreamResult(outputFile);
        transformer.transform(source, outFile);
        System.out.println("Sikeres kiírás a fájlba: " +
outputFile.getAbsolutePath());
    }

    // XML dokumentum felépítése (beolvasás és normalizálás)
    private static Document buildDocument(File xmlFile) throws
Exception {
        DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document doc = builder.parse(xmlFile);
        doc.getDocumentElement().normalize();
        return doc;
    }

    // Gépjármű információinak kiírása
    private static void kiirGepjarmuInfo(Element gepjarmu) {
        System.out.println("\n\tJelenlegi elem: Gepjarmu");
        String jarmuID = gepjarmu.getAttribute("JarmuID");
        System.out.println("Jármű ID: " + jarmuID);

        String rendszam = getTextContent(gepjarmu,
"Rendszam");
        String marka = getTextContent(gepjarmu, "Marka");
        String evjarat = getTextContent(gepjarmu, "Evjarat");
        String gyartasiHely = getTextContent(gepjarmu,
"GyartasiHely");
        String tipus = getTextContent(gepjarmu, "Tipus");
        String megtettKm = getTextContent(gepjarmu,
"MegtettKilometer");

        System.out.println("Rendszám: " + rendszam);
        System.out.println("Márka: " + marka);
```

```
        System.out.println("Évjárat: " + evjarat);
        System.out.println("Gyártási hely: " + gyartasiHely);
        System.out.println("Típus: " + tipus);
        System.out.println("Megtett kilométer: " + megtettKm);
    }

    // EgyediNyilvantartas információinak kiírása
    private static void kiirEgyediNyilvantartasInfo(Element
egyedi) {
        System.out.println("\n\tJelenlegi elem:
EgyediNyilvantartas");
        String jarmuID = egyedi.getAttribute("JarmuID");
        System.out.println("Jármű ID (hivatkozás): " +
jarmuID);

        String rendszam = getTextContent(egyedi, "Rendszam");
        String kgfb = getTextContent(egyedi, "KGFB");
        String tulajdonos = getTextContent(egyedi,
"Tulajdonos");

        System.out.println("Rendszám: " + rendszam);
        System.out.println("KGFB: " + kgfb);
        System.out.println("Tulajdonos: " + tulajdonos);
    }

    // Vásárló információinak kiírása
    private static void kiirVasarloInfo(Element vasarlo) {
        System.out.println("\n\tJelenlegi elem: Vasarlo");
        String vasarloID = vasarlo.getAttribute("VasarloID");
        System.out.println("Vásárló ID: " + vasarloID);

        String nev = getTextContent(vasarlo, "Nev");
        String lakcim = getTextContent(vasarlo, "Lakcim");
        String telefonszam = getTextContent(vasarlo,
"Telefonszam");

        System.out.println("Név: " + nev);
        System.out.println("Lakcím: " + lakcim);
        System.out.println("Telefonszám: " + telefonszam);
    }

    // Kölcsönzés információinak kiírása
    private static void kiirKolcsonzesInfo(Element kolcsonzes)
{
        System.out.println("\n\tJelenlegi elem: Kolcsonzes");
```

```
        String jarmuID =
kolcsonzes.getAttribute("GepjarmuID");
        String vasarloID =
kolcsonzes.getAttribute("VasarloID");
        System.out.println("Gépjármű ID (ref): " + jarmuID);
        System.out.println("Vásárló ID (ref): " + vasarloID);

        String tol = getTextContent(kolcsonzes, "Tol");
        String ig = getTextContent(kolcsonzes, "Ig");
        String elteltNapok = getTextContent(kolcsonzes,
"ElteltNapok");
        String kolcsonzesiAr = getTextContent(kolcsonzes,
"KolcsonzesiAr");

        System.out.println("Bérlés - Tól: " + tol);
        System.out.println("Bérlés - Ig: " + ig);
        System.out.println("Eltelt napok: " + elteltNapok);
        System.out.println("Kölcsönzési ár: " +
kolcsonzesiAr);
    }

    // Szerelő információinak kiírása
    private static void kiirSzereloInfo(Element szerelo) {
        System.out.println("\n\tJelenlegi elem: Szerelo");
        String szereloID = szerelo.getAttribute("SzereloID");
        System.out.println("Szerelő ID: " + szereloID);

        String nev = getTextContent(szerelo, "Nev");
        String lakcim = getTextContent(szerelo, "Lakcim");
        String telefonszam = getTextContent(szerelo,
"Telefonszam");

        System.out.println("Név: " + nev);
        System.out.println("Lakcím: " + lakcim);
        System.out.println("Telefonszám: " + telefonszam);
    }

    // Kölcsönző Cég információinak kiírása
    private static void kiirCegInfo(Element ceg) {
        System.out.println("\n\tJelenlegi elem: KolcsonoCeg");
        String cegID = ceg.getAttribute("CegID");
        System.out.println("Cég ID: " + cegID);

        String cim = getTextContent(ceg, "Cim");
        String alapterulet = getTextContent(ceg,
"Alapterulet");
```

```
            String autokSzama = getTextContent(ceg, "AutokSzama");

            System.out.println("Cím: " + cim);
            System.out.println("Alapterület: " + alapterulet);
            System.out.println("Autók száma: " + autokSzama);
        }

        // Karbantartás információinak kiírása
        private static void kiirKarbantartasInfo(Element karb) {
            System.out.println("\n\tJelenlegi elem:
Karbantartas");
            String jarmuID = karb.getAttribute("GepjarmuID");
            String szereloID = karb.getAttribute("SzereloID");
            System.out.println("Gépjármű ID (ref): " + jarmuID);
            System.out.println("Szerelő ID (ref): " + szereloID);

            String datum = getTextContent(karb, "Datum");
            String megtettKm = getTextContent(karb,
"MegtettKilometer");

            System.out.println("Dátum: " + datum);
            System.out.println("Megtett kilométer a
karbantartáskor: " + megtettKm);
        }

        // Dolgozik információinak kiírása
        private static void kiirDolgozikInfo(Element dolgozik) {
            System.out.println("\n\tJelenlegi elem: Dolgozik");
            String szereloID = dolgozik.getAttribute("SzereloID");
            String cegID = dolgozik.getAttribute("CegID");
            System.out.println("Szerelő ID (ref): " + szereloID);
            System.out.println("Cég ID (ref): " + cegID);

            String kezdoDatum = getTextContent(dolgozik,
"KezdoDatum");
            System.out.println("Kezdődátum: " + kezdoDatum);
        }

        // Kiadva információinak kiírása
        private static void kiirKiadvaInfo(Element kiadva) {
            System.out.println("\n\tJelenlegi elem: Kiadva");
            String jarmuID = kiadva.getAttribute("GepjarmuID");
            String cegID = kiadva.getAttribute("CegID");
            System.out.println("Gépjármű ID (ref): " + jarmuID);
            System.out.println("Cég ID (ref): " + cegID);
```

```
            String datum = getTextContent(kiadva, "Datum");
            System.out.println("Kiadás dátuma: " + datum);
        }

        // Segédfüggvény a szövegtartalom lekérésére
        private static String getTextContent(Element parent,
String tagName) {
            Node node =
parent.getElementsByTagName(tagName).item(0);
            if (node != null) {
                return node.getTextContent();
            }
            return "";
        }
}
```

2.2 DOM adatírás

DomWriteNLFUA8:

Ez a kód egy **XML dokumentum generáló és kiíró program** Java nyelven, amely az **autókölcsönzés rendszeréhez** kapcsolódó adatokat hoz létre. Az adatok egy hierarchikus XML szerkezetben jelennek meg, különféle entitásokkal, mint például gépjárművek, vásárlók, kölcsönzések, szerelők, cégek stb.

Áttekintés

1. **Main metódus**:
    - A createAndPrintXMLDocument() metódust hívja meg, amely a dokumentumot létrehozza és kiírja.
    - Hiba esetén az üzenetet a konzolra írja.
2. **XML dokumentum létrehozása**:
    - Az XML dokumentum alapját a Java DOM API (DocumentBuilderFactory, DocumentBuilder) biztosítja.
    - Az alapértelmezett gyökérelem az AutokolcsonzesRendszer, amely egy xsi:noNamespaceSchemaLocation attribútummal van ellátva.
3. **Adatok hozzáadása**:
    - Az adatok hozzáadására külön metódusok vannak, például:
        - addGepjarmu: Gépjármű adatok (pl. rendszám, márka, évjárat).
        - addVasarlo: Vásárlók adatai (pl. név, lakcím).

        - addKolcsonzes: Kölcsönzési tranzakciók.
        - Ezek a metódusok XML elemeket hoznak létre, majd a gyökérelemhez csatolják őket.

4. **Adatok kiírása**:
    - A printDocumentToConsole() metódus a konzolra írja az XML-t szép formázással.
    - A printToFile() metódus fájlba írja az XML-t egy megadott útvonalon.

Fontosabb részek részletesen

XML dokumentum struktúra

A kód egy komplex XML dokumentumot generál, amely több szinten tartalmaz adatokat:

- **Gyökérelem**: AutokolcsonzesRendszer
    - Gyermekelemek:
        - Gepjarmu: Gépjárművek listája.
        - EgyediNyilvantartas: Gépjármű nyilvántartási adatai.
        - Vasarlo: Vásárlók.
        - Kolcsonzes: Kölcsönzési adatok.
        - Szerelo: Szerelők.
        - KolcsonoCeg: Kölcsönző cégek.
        - Karbantartas: Karbantartási adatok.
        - Dolgozik: Szerelők és cégek kapcsolata.
        - Kiadva: Kölcsönzött gépjárművek.

Az XML szerkezet jól reprezentálja az autókölcsönzési rendszer entitásait és azok kapcsolatait.

Egyedi metódusok

Minden adatcsoporthoz dedikált metódus tartozik, amely biztosítja:

- Az adatok struktúrált hozzáadását.
- Attribútumok beállítását (pl. JarmuID, SzereloID).
- Gyermekelemek generálását (pl. Rendszam, Marka a Gepjarmu alatt).

Fontosnak tartott kódrészlet:

addGepjarmu()

Jellemzői:

- Egy Gepjarmu elemet hoz létre és tölti fel adatokkal.
- appendElement-et használ a gyermekelemek hozzáadására, ami a kódot újrahasznosíthatóvá teszi.

Általános segédmetódus

Az appendElement metódus az egyik legfontosabb:

```
private static void appendElement(Document doc, Element
parent, String tagName, String text) {

    Element elem = doc.createElement(tagName);

    elem.appendChild(doc.createTextNode(text));

    parent.appendChild(elem);

}
```

Ez a metódus:

- Csökkenti a kódismétlést.
- Automatikusan létrehozza a megadott névvel (tagName) az elemet, és hozzáadja a szöveges tartalmat (text).

Használata: Több helyen is alkalmazzák, például a addGepjarmu metódusban.

2.3 DOM adatlekérdezés

DomQueryNLFUA8:

A kód egy Java program, amely DOM (Document Object Model) API-t használ XML fájl feldolgozására és különböző lekérdezések végrehajtására. A fő cél az XML adatok strukturált módon történő beolvasása és különböző szűrések végrehajtása. Nézzük részletesen a programot!

Kódfelépítés

1. **Csomag deklaráció és importálás**
    - A hu.domparse.nlfua8 csomag alatt található a program.

- Az org.w3c.dom és javax.xml.parsers osztályokat importálja DOM API használatához.
- A java.io.File osztály a fájlkezeléshez szükséges.

2. **Főmetódus (main)**
   - A main metódus az XML fájl útvonalát adja meg: src/resources/XMLNLFUA8.xml.
   - Létrehoz egy DOM dokumentumot a fájlból.
   - Normalizálja az XML dokumentumot (doc.getDocumentElement().normalize()), hogy a whitespace-ek kezelése egységes legyen.
   - Öt különböző lekérdezést hajt végre az XML dokumentumon keresztül, mindegyik egy külön metódusban van implementálva.
3. **Egyes metódusok részletezése**

1. listAllVehicles

- **Cél:** Az összes gépjármű adatainak listázása.
- **XML tag-ek:** <Gepjarmu>, <Rendszam>, <Marka>.
- Az getElementsByTagName metódust használja a gépjárművek lekérésére.
- Az egyes gépjárművek attribútumait (JarmuID) és gyermekeit (Rendszam, Marka) olvassa ki.

2. listAllCustomers

- **Cél:** Az összes vásárló listázása.
- **XML tag-ek:** <Vasarlo>, <Nev>, <Telefonszam>.
- Az getElementsByTagName metódussal a <Vasarlo> elemeket gyűjti be.
- Az attribútum (VasarloID) és gyermekelemek (Nev, Telefonszam) kiolvasása történik.

3. listAllRentalsAbovePrice

- **Cél:** Az összes olyan kölcsönzés listázása, amelynek ára meghalad egy adott küszöbértéket.
- **Paraméter:** Árküszöb (priceThreshold).
- **XML tag-ek:** <Kolcsonzes>, <KolcsonzesiAr>, <Tol>, <Ig>, <ElteltNapok>.
- Az ár (KolcsonzesiAr) összehasonlítására a Double.parseDouble metódust használja.

- Az eredményeket csak akkor írja ki, ha az ár nagyobb a megadott küszöbértéknél.

4. listAllMaintenancesBeforeDate

- **Cél:** Az összes karbantartás listázása, amely egy adott dátum előtt történt.
- **Paraméter:** Dátumhatár (dateLimit).
- **XML tag-ek:** <Karbantartas>, <Datum>, <MegtettKilometer>.
- A dátumokat lexikografikusan hasonlítja össze (datum.compareTo(dateLimit) < 0).

5. listAllWorkingRelations

- **Cél:** Minden "Dolgozik" kapcsolat listázása.
- **XML tag-ek:** <Dolgozik>, <KezdoDatum>.
- Az attribútumokat (SzereloID, CegID) és gyermekelemeket (KezdoDatum) kiolvasva listázza a kapcsolatokat.

Fontosabb részletek

1. **Normalizálás:**
   - A doc.getDocumentElement().normalize() biztosítja, hogy az XML dokumentum whitespace-jei egyenletesen legyenek kezelve. Ez különösen fontos, ha a dokumentumot emberi kéz is szerkesztette, vagy nem minden adat egy sorban van.
2. **Attribútumok és gyermekelemek kezelése:**
   - Az Element.getAttribute() metódussal az XML attribútumokat olvassa ki.
   - Az getElementsByTagName segítségével gyermekelemek tartalmát nyeri ki.
3. **Hibakezelés:**
   - Az egész program egy try-catch blokkba van csomagolva, hogy a lehetséges kivételeket (pl. fájl hiánya vagy rossz XML formátum) kezelni tudja.

Fontosnak tartott kódrészlet:

listAllRentalsAbovePrice()

A **listAllRentalsAbovePrice() metódus** kiemelkedik, mert jól demonstrálja, hogyan használhatunk dinamikus szűrőfeltételeket egy XML dokumentumon. Az alábbi szempontok miatt érdekes:

- Dinamikus paraméterezés (priceThreshold).
- A numerikus adatok konvertálása és összehasonlítása (Double.parseDouble).

- Több attribútum és gyermekelem kezelése.
- Példa arra, hogyan lehet az XML adatokkal feltételes logikát megvalósítani.

2.4 DOM adatmódosítás

DomModifyNLFUA8:

A kódot részletesen elemezzük, figyelve az alkalmazott technológiákra, a logika felépítésére és az esetleges hibalehetőségekre.

Általános áttekintés

- A kód egy XML-fájl feldolgozására készült. Az XML-t a DOM API segítségével módosítja, ami a teljes fájl betöltésén alapul, lehetővé téve a csomópontok könnyű elérését és manipulálását.
- A kód négy különböző módosítást végez a betöltött XML-en:
    1. **Kilométeróra állás növelése egy adott gépjárműnél.**
    2. **Új karbantartás rekord hozzáadása.**
    3. **Egy vásárló telefonszámának frissítése.**
    4. **Kölcsönzési ár módosítása.**

Elemek részletes elemzése

Fájl betöltése és feldolgozás

```
File xmlFile = new File("src/resources/XMLNLFUA8.xml");

DocumentBuilder builder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();

Document doc = builder.parse(xmlFile);

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

Az XML-fájl betöltése biztonságosan történik a DocumentBuilderFactory és a DocumentBuilder segítségével.

A .normalize() hívás biztosítja, hogy a DOM-fa konzisztens legyen (pl. eltávolítja az üres csomópontokat).

Módosítások naplózása

```
StringBuilder changesLog = new StringBuilder();
```

A változásokat egy StringBuilder tárolja, amelyet a végén kiírnak. Ez a megoldás egyszerű, de hatékony a nyomon követéshez.

Kilométeróra növelése (CAR1)

```
NodeList cars = doc.getElementsByTagName("Gepjarmu");

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

     Element car = (Element) cars.item(i);

          if (car.getAttribute("JarmuID").equals("CAR1")) {

          Node kmNode =
car.getElementsByTagName("MegtettKilometer").item(0);

          int currentKm =
Integer.parseInt(kmNode.getTextContent());

          kmNode.setTextContent(String.valueOf(currentKm +
1000));

          changesLog.append("Módosított CAR1 kilométeróra: ")

          .append(kmNode.getTextContent()).append("\n");

          }

}
```

Működés:

Kiválasztja a Gepjarmu csomópontokat, majd az attribútumok alapján ellenőrzi, hogy a megfelelő gépjárműnél van-e.

Új karbantartási rekord hozzáadása

```
Element newMaintenance = doc.createElement("Karbantartas");
newMaintenance.setAttribute("GepjarmuID", "CAR2");
newMaintenance.setAttribute("SzereloID", "SZ1");

Element datum = doc.createElement("Datum");
datum.setTextContent(LocalDate.now().toString());
newMaintenance.appendChild(datum);

Element km = doc.createElement("MegtettKilometer");
km.setTextContent("27000");
newMaintenance.appendChild(km);

Node root = doc.getDocumentElement();
```

```
root.appendChild(newMaintenance);
```

Működés:

Egy új DOM-csomópontot hoz létre, amely tartalmazza a szükséges elemeket és attribútumokat.

Vásárló telefonszámának frissítése

```
NodeList vasarlok = doc.getElementsByTagName("Vasarlo");
for (int i = 0; i < vasarlok.getLength(); i++) {
    Element vasarlo = (Element) vasarlok.item(i);
    if (vasarlo.getAttribute("VasarloID").equals("V1")) {
        Node telefonNode =
vasarlo.getElementsByTagName("Telefonszam").item(0);
        telefonNode.setTextContent("9999999");
        changesLog.append("Frissített telefonszám V1-hez: ")

.append(telefonNode.getTextContent()).append("\n");
    }
}
```

- **Működés:** Frissíti a megadott vásárló Telefonszam mezőjét.
- **Hibalehetőség:** A vasarlo.getElementsByTagName("Telefonszam").item(0) null értéket adhat vissza, ha nincs ilyen elem. Ezt kezelni kellene.

Kölcsönzési ár növelése

```
NodeList kolcsonzesek = doc.getElementsByTagName("Kolcsonzes");
for (int i = 0; i < kolcsonzesek.getLength(); i++) {
    Element kolcsonzes = (Element) kolcsonzesek.item(i);
    if (kolcsonzes.getAttribute("GepjarmuID").equals("CAR3") &&
        kolcsonzes.getAttribute("VasarloID").equals("V3")) {

        Node arNode =
kolcsonzes.getElementsByTagName("KolcsonzesiAr").item(0);
        double currentPrice =
Double.parseDouble(arNode.getTextContent());
        arNode.setTextContent(String.valueOf(currentPrice +
50.00));
        changesLog.append("Módosított kölcsönzési ár CAR3 és V3
esetén: ")
                  .append(arNode.getTextContent()).append("\n");
    }
}
```

Működés:

Kiválasztja a megfelelő kölcsönzést, majd frissíti az árat.

Módosítások kiírása

```
private static void printModifiedXml(String changes) {
    System.out.println("-------Módosítások--------");
    System.out.println(changes);
}
```

Fontosnak tartott kódrészlet

Elérhető itt.

A **kilométeróra módosítása** fontos, mert egy egyszerű, de tipikus XML-manipulációs művelet:

```
NodeList cars = doc.getElementsByTagName("Gepjarmu");
for (int i = 0; i < cars.getLength(); i++) {
    Element car = (Element) cars.item(i);
    if (car.getAttribute("JarmuID").equals("CAR1")) {
        Node kmNode =
car.getElementsByTagName("MegtettKilometer").item(0);
        int currentKm =
Integer.parseInt(kmNode.getTextContent());
        kmNode.setTextContent(String.valueOf(currentKm + 1000));
    }
}
```