JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Repülőtér

Készítette: Hauer Attila Árpád

Neptun kód: JJL4WE

Gyakorlatvezető: Dr. Bednarik László

Gyakorlat időpontja: Szerda 1200-14:00

Képzés: Mérnökinformatikus BSc nappali

Tartalomjegyzék

1. Feladat leírása

A féléves beadandóm tematikája egyrepülőtér amelynek lehet látni induló repülőit, légitársaságait XML-ben. Az XML dokumentumban tudunk beolvasni, írni, lekérdezni és módosítani.

1a.) Az egyedek tulajdonságai

- **Járat:**
  - **Légitársaság_ID:** A Járat egyed elsődleges kulcsa.
  - **Státusz:** Nyilvántartja a státuszát, hogy késik e vagy felszállásra kész.
  - **Induási_idő:** A repülőgép indulási ideje.
  - **Érkezési_idő:** A repülőgép érkezési ideje.
  - **Célállomás:** A repülőgép úticélja. (VÁROS)

- **Jegy:**
  - **Jegysorszám:** A Jegy egyed elsődleges kulcsa.
  - **Ár:** A jegy ára.
  - **Úticél:** Összetett tulajdonság, amely tartalmazza a reptér nevét, illetve országát.
  - **Induási_idő:** A repülőgép indulási ideje.
  - **Érkezési_idő:** A repülőgép érkezési ideje.
  - **Hely:** Összetett tulajdonság, amely tartalmazza a székszámot, illetve az oszályt.

- **Utasok:**
  - **Jegysorszám:** A Utasok egyed elsődleges kulcsa.
  - **Név:** Az utas neve.
  - **Születésiév:** Az utas születési dátuma.
  - **Nem:** Az utas neme.
  - **Lakcím:** Az utas lakcíme.
  - **Telefonszám:** Az utas telefonszáma. Többértékű tulajdonság hisz, egy utasnak lehet több telefonszáma.

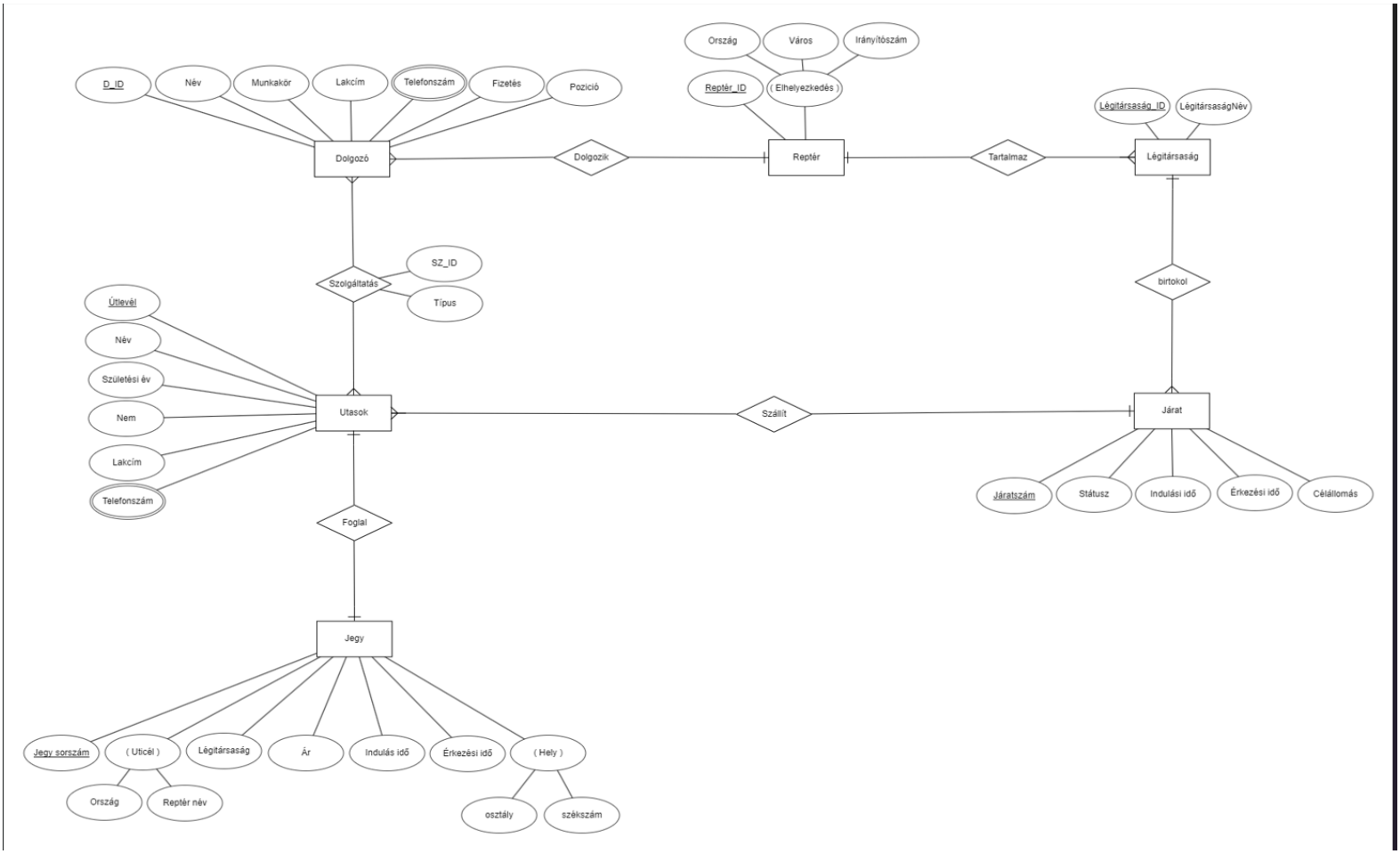
- **Szolgáltatás:**
  - **SZ_ID:** A szolgáltatás kapcsolat elsődleges kulcsa.
  - **Típus:** Szolgáltatás típusa

- **Dolgozó:**
  - **D_ID:** A Dolgozó egyed elsődleges kulcsa.
  - **Név:** Az dolgozó neve.
  - **Munkakör:** A dolgozó munkaköre.
  - **Lakcím:** Az dolgozó lakcíme.
  - **Telefonszám:** Az dolgozó telefonszáma. Többértékű tulajdonság hisz, egy dolgozónak több telefonszáma is lehet.
  - **Fizetés:** A dolgozó fizetése.
  - **Pozíció:** A dolgozó hierarchikus pozíciója

- **Reptér:**
  - **Reptér_ID:** A Reptér egyed elsődleges kulcsa.
  - **Elhelyezkedés:** Összetett tulajdonság, amelynek van ország, város és irányítószám tulajdonosa.

- **Légitársaság:**
  - **Légitársaság_ID:** A Légitársaság egyed elsődleges kulcsa.
  - **Név:** A mérkőzés típusának a neve. Lehet csoport-, nyolcad-, negyed-, elő- és döntő.

1b.) A feladat ER Modellje



1c.) Az egyedek közötti kapcsolatok

- **Járat és Utasok:**
  - A **Járat** és az **Utasok** között **1:N** típusú kapcsolat van, mivel egy járaton több utas is utazhat, de egy utas egyszerre csak egy járaton utazhat

- **Utasok és Dolgozó:**
  - A **Dolgozó** és az **Utasok** között **N:N** típusú kapcsolat van, mivel egy dolgozó több féle szolgáltatást nyújthat egy utas számára, de egy utas is kérhet több szolgáltatást.

- **Reptér és Dolgozó:**
  - A **Reptér** és a **Dolgozó** között **1:N** típusú kapcsolat van, egy reptéren több dolgozó dolgozik, de egy dolgozó csak egy reptéren dolgozhat.

- **Reptér és Légitársaság:**
  - A **Reptér** és a **Légitársaság** között **1:N** típusú kapcsolat van, mivel egy reptéren lehet több légitársaság, de egy légitársaság csak egy reptérhez tartozhat.

- **Légitársaság és Járat:**
  - A **Légitársaság** és a **Járat** között **1:N** típusú kapcsolat van, mivel egy légitársaságnak több járata is lehet, de egy járat egy adott legitársasághoz tartozik.

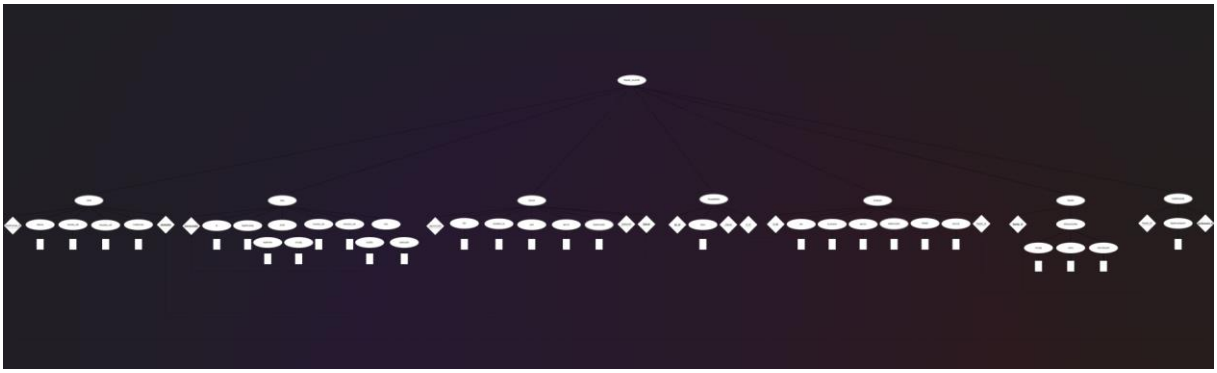
- **Utasok és Jegy:**
  - Az **Utasok** és a **Jegy** között **1:1** típusú kapcsolat van, mivel egy utashoz egy jegy tartozik és egy jegyet egy utas birtokol.

2. Az ER modell konvertálása XDM modellre

A jelölésrendszerünk háromféle szimbólumot alkalmaz a strukturális elemek, attribútumok és szöveges tartalom megkülönböztetésére. Az ellipszis szimbólumot használjuk az elemek ábrázolására, ami minden egyes adatbáziselemet és tulajdonságot magába foglal. A rombusz jelképezi az attribútumokat, amelyek a kulcs tulajdonságokból származnak. A téglalapok pedig a szöveges tartalmat reprezentálják, és ezek a részek fogják alkotni az XML dokumentumot.

Azok az elemek, amelyek többször is megjelenhetnek, két ellipszissel vannak ábrázolva. Az idegenkulcsok és kulcsok közötti kapcsolatot szaggatott vonalas nyíllal jelöljük, hangsúlyozva ezen elemek közötti kapcsolat fontosságát.

XDM modell:



3. XML dokumentum készítése

Az XDM modell alapján készítettem el az XML dokumentumot. Legelőször a gyökérelemmel kezdtem, amelynek a **„Reptér_JJL4WE”** nevet adtam. Ezek után a gyermekelemeiből eltérő módon hoztam létre példányokat.

XML dokumentum forráskódja:

```
<?xml version="1.0" encoding="UTF-8"?>
<Reptér_JJL4WE xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="XMLSchemaJJL4WE.xsd">

    <!--Járat-->
    <Járat Járatszám="111" Légitársaság_ID="12">
        <célállomás>Budapest</célállomás>
```

```
        <indulási_idő>2023/11/17 17:00</indulási_idő>
        <érkezési_idő>2023/11/17 19:00</érkezési_idő>
        <státusz>késik</státusz>
    </Járat>

    <Járat Járatszám="112" Légitársaság_ID="11">
        <célállomás>Atlanta</célállomás>
        <indulási_idő>2023/11/20 11:00</indulási_idő>
        <érkezési_idő>2023/11/20 23:00</érkezési_idő>
        <státusz>várható</státusz>

    </Járat>

    <Járat Járatszám="113" Légitársaság_ID="13">
        <célállomás>London</célállomás>
        <indulási_idő>2023/11/18 23:00</indulási_idő>
        <érkezési_idő>2023/11/19 06:00</érkezési_idő>
        <státusz>várható</státusz>
    </Járat>

    <!--Jegy-->
    <Jegy Jegysorszám="1111">
        <ár>40000</ár>
        <légitársaság>Emirates</légitársaság>
        <uticél>
            <reptérnév>Heathrow</reptérnév>
            <ország>Egyesült Királyság</ország>
        </uticél>
        <indulási_idő>2023/11/18 23:00</indulási_idő>
        <érkezési_idő>2023/11/19 06:00</érkezési_idő>
        <hely>
            <osztály>1</osztály>
            <székszám>31</székszám>
        </hely>
    </Jegy>

    <Jegy Jegysorszám="1112">
        <ár>20000</ár>
        <légitársaság>Qatar Airways</légitársaság>
        <uticél>
            <reptérnév>Ferihegy</reptérnév>
            <ország>Magyarország</ország>
        </uticél>
        <indulási_idő>2023/11/17 17:00</indulási_idő>
        <érkezési_idő>2023/11/17 19:00</érkezési_idő>
        <hely>
            <osztály>2</osztály>
            <székszám>52</székszám>
        </hely>
```

```xml
</Jegy>

<Jegy Jegysorszám="1113">
    <ár>50000</ár>
    <légitársaság>Lufthansa</légitársaság>
    <uticél>
        <reptérnév>Hartsfield-Jackson</reptérnév>
        <ország>Egyesült Államok</ország>
    </uticél>
    <indulási_idő>2023/11/20 11:00</indulási_idő>
    <érkezési_idő>2023/11/20 23:00</érkezési_idő>
    <hely>
        <osztály>1</osztály>
        <székszám>11</székszám>
    </hely>
</Jegy>

<!--Utasok-->
<Utasok Útlevél="98765432" Jegysorszám="1113" Járatszám="113">
    <név>Hauer Attila</név>
    <születésiév>2002</születésiév>
    <nem>Férfi</nem>
    <lakcím>Magyarország, 3527 Miskolc József Attila u. 12</lakcím>
    <telefonszám>+36201234567</telefonszám>
    <telefonszám>+36701234567</telefonszám>
</Utasok>

<Utasok Útlevél="87654367" Jegysorszám="1112" Járatszám="111">
    <név>Olivia Thompson</név>
    <születésiév>2001</születésiév>
    <nem>Nő</nem>
    <lakcím>Nagy-Britannia, SW1A 2AB London Downing Street 10</lakcím>
    <telefonszám>+36209876428</telefonszám>
    <telefonszám>+36702138757</telefonszám>
    <telefonszám>+36705834653</telefonszám>
</Utasok>

<Utasok Útlevél="87575645" Jegysorszám="1111" Járatszám="112">
    <név>Benjamin Mitchell</név>
    <születésiév>1970</születésiév>
    <nem>Nő</nem>
    <lakcím>USA, CA 90212 Beverly Hills Rodeo Drive 123</lakcím>
    <telefonszám>+36202345965</telefonszám>
</Utasok>

<!--Szolgáltatás (Dolgozó-Utasok kapcsolat)-->
<Szolgáltatás SZ_ID="3001" D_ID="2001" Útlevél="98765432">
    <típus>Információ nyújtás</típus>
</Szolgáltatás>
```

```xml
<Szolgáltatás SZ_ID="3002" D_ID="2002" Útlevél="87575645">
    <típus>Felszolgálás</típus>
</Szolgáltatás>

<Szolgáltatás SZ_ID="3003" D_ID="2003" Útlevél="87654367">
    <típus>Bőrönd mérés</típus>
</Szolgáltatás>

<!--Dolgozó-->
<Dolgozó D_ID="2001" Reptér_ID="1">
    <név>Balla Sándor</név>
    <munkakör>Biztonságiőr</munkakör>
    <lakcím>Magyarország, 3530 Miskolc Arany János u. 32</lakcím>
    <telefonszám>+36204567534</telefonszám>
    <fizetés>450000</fizetés>
    <pozício>Biztonsági vezető</pozício>
</Dolgozó>

<Dolgozó D_ID="2002" Reptér_ID="3">
    <név>Kobe Briant</név>
    <munkakör>Felszolgáló</munkakör>
    <lakcím>USA, CA 90212 Beverly Hills Rodeo Drive 25</lakcím>
    <telefonszám>+36203641243</telefonszám>
    <fizetés>750000</fizetés>
    <pozício>Légi forgalmi irányító</pozício>
</Dolgozó>

<Dolgozó D_ID="2003" Reptér_ID="2">
    <név>Charles Hamilton</név>
    <munkakör>bőröndmérő</munkakör>
    <lakcím>Nagy-Britannia, SW1A 2AD London Downing Street 76</lakcím>
    <telefonszám>+36301237659</telefonszám>
    <telefonszám>+36708646856</telefonszám>
    <telefonszám>+36207363457</telefonszám>
    <fizetés>350000</fizetés>
    <pozício>Utasellátó</pozício>
</Dolgozó>

<!--Reptér-->
<Reptér Reptér_ID="1" >
    <elhelyezkedés>
        <ország>Magyarország</ország>
        <város>Budapest</város>
        <irányítószám>1185</irányítószám>
    </elhelyezkedés>
</Reptér>

<Reptér Reptér_ID="2" >
```

```
        <elhelyezkedés>
            <ország>Egyesült Királyság</ország>
            <város>London</város>
            <irányítószám>TW62GA</irányítószám>
        </elhelyezkedés>
    </Reptér>

    <Reptér Reptér_ID="3" >
    <elhelyezkedés>
        <ország>Egyesült Államok</ország>
        <város>Atlanta</város>
        <irányítószám>30337</irányítószám>
    </elhelyezkedés>
    </Reptér>

    <!--Légitársaság-->
    <Légitársaság Légitársaság_ID="11" Reptér_ID="1">
        <légitársaságnév>Lufthansa</légitársaságnév>
    </Légitársaság>

    <Légitársaság Légitársaság_ID="12" Reptér_ID="2">
        <légitársaságnév>Qatar Airways</légitársaságnév>
    </Légitársaság>

    <Légitársaság Légitársaság_ID="13" Reptér_ID="3">
        <légitársaságnév>Emirates</légitársaságnév>
    </Légitársaság>

</Reptér_JJL4WE>
```

4. XMLSchema készítése XML dokumentum alapján

Az XML dokumentumhoz készíteni kellett egy validációt elősegítő sémát. Először kigyűjtöttem az egyszerű típusokat, majd meghatároztam a saját típusokat. Összesen 4 darabot hoztam létre. Például a dátumhoz készítettem egy olyan saját típust, amely reguláris kifejezést tartalmaz, illetve enumerációt a jelöléshez. Ezek után elkészítettem a komplex típusokat minden elemre, majd elsődleges- és idegenkulcsokat hoztam létre. A legvégén pedig megvalósítottam az 1:1 kapcsolatot a Jegy és az Utasok egyedek között.

XMLSchema forráskódja:

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

<!--Egyszerű típusok kigyűjtése-->
    <!--Járat-->
    <xs:element name="célállomás" type="xs:string" />
    <xs:element name="érkezési_idő" type="idoTipus" />
```

```
    <xs:element name="indulási_idő" type="idoTipus" />
    <xs:element name="státusz" type="xs:string" />

    <!--Jegy-->
    <xs:element name="ár" type="xs:int"/>
    <xs:element name="légitársaság" type="xs:string" />
    <xs:element name="reptérnév" type="xs:string" />
    <xs:element name="uticél"/>
    <xs:element name="hely"/>
    <xs:element name="osztály" type="osztály" />
    <xs:element name="székszám" type="xs:int" />

    <!--Utasok-->
    <xs:element name="név" type="xs:string" />
    <xs:element name="nem" type="nemTípus" />
    <xs:element name="lakcím" type="xs:string" />
    <xs:element name="telefonszám" type="telefonTípus" />
    <xs:element name="születésiév" type="xs:int" />

    <!--Szolgáltatás (Dolgozó-Utasok kapcsolat)-->
    <xs:element name="típus" type="xs:string" />

    <!--Dolgozó-->
    <xs:element name="munkakör" type="xs:string" />
    <xs:element name="fizetés" type="xs:int" />
    <xs:element name="pozício" type="xs:string" />

    <!--Reptér-->
    <xs:element name="elhelyezkedés"/>
    <xs:element name="ország" type="xs:string" />
    <xs:element name="város" type="xs:string" />
    <xs:element name="irányítószám" type="xs:string" />

    <!--Légitársaság-->
    <xs:element name="légitársaságnév" type="xs:string" />


    <!--Saját típus meghatározások-->
    <xs:simpleType name="idoTipus">
        <xs:restriction base="xs:string">
            <xs:pattern value="\d{4}/\d{2}/\d{2} \d{2}:\d{2}" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="osztály">
        <xs:restriction base="xs:int">
            <xs:minInclusive value="1"></xs:minInclusive>
            <xs:maxInclusive value="2"></xs:maxInclusive>
        </xs:restriction>
```

```
    </xs:simpleType>

    <xs:simpleType name="nemTípus">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Férfi"></xs:enumeration>
            <xs:enumeration value="Nő"></xs:enumeration>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="telefonTípus">
        <xs:restriction base="xs:string">
            <xs:pattern value="\+\d{1,11}" />
        </xs:restriction>
    </xs:simpleType>



    <!--Komplex típusok meghatározása-->
    <xs:complexType name="járatTípus">
        <xs:sequence>
            <xs:element ref="célállomás"/>
            <xs:element ref="indulási_idő"/>
            <xs:element ref="érkezési_idő"/>
            <xs:element ref="státusz"/>
        </xs:sequence>
        <xs:attribute name="Járatszám" type="xs:int"/>
        <xs:attribute name="Légitársaság_ID" type="xs:int"/>
    </xs:complexType>

    <xs:complexType name="jegyTípus">
        <xs:sequence>
            <xs:element ref="ár"/>
            <xs:element ref="légitársaság"/>
            <xs:element name="uticél">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="reptérnév"/>
                        <xs:element ref="ország"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element ref="indulási_idő"/>
            <xs:element ref="érkezési_idő"/>
            <xs:element name="hely">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="osztály"/>
                        <xs:element ref="székszám"/>
                    </xs:sequence>
```

```
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="Jegysorszám" type="xs:int"/>
    </xs:complexType>

    <xs:complexType name="utasokTípus">
        <xs:sequence>
            <xs:element ref="név"/>
            <xs:element ref="születésiév"/>
            <xs:element ref="nem"/>
            <xs:element ref="lakcím"/>
            <xs:element name="telefonszám" type="telefonTípus" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Jegysorszám" type="xs:int"/>
        <xs:attribute name="Járatszám" type="xs:int"/>
        <xs:attribute name="Útlevél" type="xs:int"/>
    </xs:complexType>

    <xs:complexType name="szolgáltatásTípus">
        <xs:sequence>
            <xs:element ref="típus"/>
        </xs:sequence>
        <xs:attribute name="SZ_ID" type="xs:int"/>
        <xs:attribute name="Útlevél" type="xs:int"/>
        <xs:attribute name="D_ID" type="xs:int"/>
    </xs:complexType>

    <xs:complexType name="dolgozóTípus">
        <xs:sequence>
            <xs:element ref="név"/>
            <xs:element ref="munkakör"/>
            <xs:element ref="lakcím"/>
            <xs:element name="telefonszám" type="telefonTípus" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element ref="fizetés"/>
            <xs:element ref="pozício"/>
        </xs:sequence>
        <xs:attribute name="Reptér_ID" type="xs:int"/>
        <xs:attribute name="D_ID" type="xs:int"/>
    </xs:complexType>

    <xs:complexType name="reptérTípus">
        <xs:sequence>
            <xs:element name="elhelyezkedés">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="ország"/>
```

```
                    <xs:element ref="város"/>
                    <xs:element ref="irányítószám"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="Reptér_ID" type="xs:int"/>
</xs:complexType>

<xs:complexType name="légitársaságTípus">
    <xs:sequence>
        <xs:element name="légitársaságnév"/>
    </xs:sequence>
    <xs:attribute name="Reptér_ID" type="xs:int"/>
    <xs:attribute name="Légitársaság_ID" type="xs:int"/>
</xs:complexType>

<xs:element name="Reptér_JJL4WE">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Járat" type="járatTípus" minOccurs="1"
maxOccurs="unbounded"/>
            <xs:element name="Jegy" type="jegyTípus" minOccurs="1"
maxOccurs="unbounded"/>
            <xs:element name="Utasok" type="utasokTípus" minOccurs="1"
maxOccurs="unbounded"/>
            <xs:element name="Szolgáltatás" type="szolgáltatásTípus"
minOccurs="1" maxOccurs="unbounded"/>
            <xs:element name="Dolgozó" type="dolgozóTípus" minOccurs="1"
maxOccurs="unbounded"/>
            <xs:element name="Reptér" type="reptérTípus" minOccurs="1"
maxOccurs="unbounded"/>
            <xs:element name="Légitársaság" type="légitársaságTípus"
minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <!--Elsődleges kulcsok-->
    <xs:key name="Járatszám_kulcs">
        <xs:selector xpath="Járat"></xs:selector>
        <xs:field xpath="@Járatszám"></xs:field>
    </xs:key>
    <xs:key name="Jegy_kulcs">
        <xs:selector xpath="Jegy"></xs:selector>
        <xs:field xpath="@Jegysorszám"></xs:field>
    </xs:key>
    <xs:key name="Utasok_kulcs">
        <xs:selector xpath="Utasok"></xs:selector>
        <xs:field xpath="@Útlevél"></xs:field>
```

```
        </xs:key>
        <xs:key name="Szolgáltatás_kulcs">
            <xs:selector xpath="Szolgáltatás"></xs:selector>
            <xs:field xpath="@SZ_ID"></xs:field>
        </xs:key>
        <xs:key name="Dolgozó_kulcs">
            <xs:selector xpath="Dolgozó"></xs:selector>
            <xs:field xpath="@D_ID"></xs:field>
        </xs:key>
        <xs:key name="Reptér_kulcs">
            <xs:selector xpath="Reptér"></xs:selector>
            <xs:field xpath="@Reptér_ID"></xs:field>
        </xs:key>
        <xs:key name="Légitársaság_kulcs">
            <xs:selector xpath="Légitársaság"></xs:selector>
            <xs:field xpath="@Légitársaság_ID"></xs:field>
        </xs:key>
        <!--Idegen kulcsok-->
        <xs:keyref name="Járat-Utasok_kulcs" refer="Járatszám_kulcs">
            <xs:selector xpath="Utasok"></xs:selector>
            <xs:field xpath="@Járatszám"></xs:field>
        </xs:keyref>
        <xs:keyref name="Jegy-Utasok_kulcs" refer="Jegy_kulcs">
            <xs:selector xpath="Utasok"></xs:selector>
            <xs:field xpath="@Jegysorszám"></xs:field>
        </xs:keyref>
        <xs:keyref name="Utasok-Szolgáltatás_kulcs" refer="Utasok_kulcs">
            <xs:selector xpath="Szolgáltatás"></xs:selector>
            <xs:field xpath="@Útlevél"></xs:field>
        </xs:keyref>
        <xs:keyref name="Dolgozó-Szolgáltatás_kulcs" refer="Dolgozó_kulcs">
            <xs:selector xpath="Szolgáltatás"></xs:selector>
            <xs:field xpath="@D_ID"></xs:field>
        </xs:keyref>
        <xs:keyref name="Reptér-Dolgozó_kulcs" refer="Reptér_kulcs">
            <xs:selector xpath="Dolgozó"></xs:selector>
            <xs:field xpath="@Reptér_ID"></xs:field>
        </xs:keyref>
        <xs:keyref name="Reptér-Légitársaság_kulcs" refer="Reptér_kulcs">
            <xs:selector xpath="Légitársaság"></xs:selector>
            <xs:field xpath="@Reptér_ID"></xs:field>
        </xs:keyref>
        <xs:keyref name="Légitársaság-Járat_kulcs" refer="Légitársaság_kulcs">
            <xs:selector xpath="Járat"></xs:selector>
            <xs:field xpath="@Légitársaság_ID"></xs:field>
        </xs:keyref>
        <!--Egy-egy kapcsolat megvalósítás-->
        <xs:unique name="Jegy-Utasok_egy_egy">
            <xs:selector xpath="Utasok"></xs:selector>
```

```
            <xs:field xpath="@Jegysorszám"></xs:field>
        </xs:unique>
    </xs:element>
</xs:schema>
```

5. DOM program készítése JAVA környezetben

A DOM programokat JAVA-ban készítettem el, ahogy a feladatkiírásban is szerepelt. Az alábbi programokat a következő alfejezetekben fogom részletesebben taglalni.

5a.) DOM adatolvasás

Adatolvasás A DOMReadJjl4we osztály beolvassa a Reptér_JJL4WE XML dokumentum tartalmát, majd kiírja azt a konzolra és egy fájlba (XMLReadjjl4we.xml) strukturált formában. Ennek az osztálynak a public minősítőjű metódusait hívja a többi osztály is, amikor XML csomópontot kell kiírniuk vagy XML dokumentumot beolvasniuk. Beolvasás után a DOM fából törlöm az üres (whitespace karaktereket tartalmazó) szöveges csomópontokat, melyek a forrás XML dokumentumban található sortörések miatt alakulnak ki. A fájlba íráshoz transformert használok, míg a konzolra íráskor lépésenként haladok végig a DOM fán, csomópontról csomópontra lépve. Output a konzolon:

```
Formázott XML:

<?xml version="1.0" encoding="UTF-8" ?>
<Reptér_JJL4WE xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="XMLSchemaJJL4WE.xsd">
    <Járat Járatszám="111" Légitársaság_ID="12">
        <célállomás>Budapest</célállomás>
        <indulási_idő>2023/11/17 17:00</indulási_idő>
        <érkezési_idő>2023/11/17 19:00</érkezési_idő>
        <státusz>késik</státusz>
    </Járat>
    <Járat Járatszám="112" Légitársaság_ID="11">
        <célállomás>Atlanta</célállomás>
        <indulási_idő>2023/11/20 11:00</indulási_idő>
        <érkezési_idő>2023/11/20 23:00</érkezési_idő>
        <státusz>várható</státusz>
    </Járat>
    <Járat Járatszám="113" Légitársaság_ID="13">
        <célállomás>London</célállomás>
        <indulási_idő>2023/11/18 23:00</indulási_idő>
        <érkezési_idő>2023/11/19 06:00</érkezési_idő>
        <státusz>várható</státusz>
    </Járat>
    <Jegy Jegysorszám="1111">
        <ár>40000</ár>
        <légitársaság>Emirates</légitársaság>
        <uticél>
            <reptérnév>Heathrow</reptérnév>
            <ország>Egyesült Királyság</ország>
        </uticél>
        <indulási_idő>2023/11/18 23:00</indulási_idő>
        <érkezési_idő>2023/11/19 06:00</érkezési_idő>
        <hely>
            <osztály>1</osztály>
            <székszám>31</székszám>
        </hely>
```

```
    </Jegy>
    <Jegy Jegysorszám="1112">
        <ár>20000</ár>
        <légitársaság>Qatar Airways</légitársaság>
        <uticél>
            <reptérnév>Ferihegy</reptérnév>
            <ország>Magyarország</ország>
        </uticél>
        <indulási_idő>2023/11/17 17:00</indulási_idő>
        <érkezési_idő>2023/11/17 19:00</érkezési_idő>
        <hely>
            <osztály>2</osztály>
            <székszám>52</székszám>
        </hely>
    </Jegy>
    <Jegy Jegysorszám="1113">
        <ár>50000</ár>
        <légitársaság>Lufthansa</légitársaság>
        <uticél>
            <reptérnév>Hartsfield-Jackson</reptérnév>
            <ország>Egyesült Államok</ország>
        </uticél>
        <indulási_idő>2023/11/20 11:00</indulási_idő>
        <érkezési_idő>2023/11/20 23:00</érkezési_idő>
        <hely>
            <osztály>1</osztály>
            <székszám>11</székszám>
        </hely>
    </Jegy>
    <Utasok Jegysorszám="1113" Járatszám="113" Útlevél="98765432">
        <név>Hauer Attila</név>
        <születésiév>2002</születésiév>
        <nem>Férfi</nem>
        <lakcím>Magyarország, 3527 Miskolc József Attila u. 12</lakcím>
        <telefonszám>+36201234567</telefonszám>
        <telefonszám>+36701234567</telefonszám>
    </Utasok>
    <Utasok Jegysorszám="1112" Járatszám="111" Útlevél="87654367">
        <név>Olivia Thompson</név>
        <születésiév>2001</születésiév>
        <nem>Nő</nem>
        <lakcím>Nagy-Britannia, SW1A 2AB London Downing Street 10</lakcím>
        <telefonszám>+36209876428</telefonszám>
        <telefonszám>+36702138757</telefonszám>
        <telefonszám>+36705834653</telefonszám>
    </Utasok>
    <Utasok Jegysorszám="1111" Járatszám="112" Útlevél="87575645">
        <név>Benjamin Mitchell</név>
        <születésiév>1970</születésiév>
        <nem>Nő</nem>
        <lakcím>USA, CA 90212 Beverly Hills Rodeo Drive 123</lakcím>
        <telefonszám>+36202345965</telefonszám>
    </Utasok>
    <Szolgáltatás D_ID="2001" SZ_ID="3001" Útlevél="98765432">
        <típus>Információ nyújtás</típus>
    </Szolgáltatás>
    <Szolgáltatás D_ID="2002" SZ_ID="3002" Útlevél="87575645">
        <típus>Felszolgálás</típus>
    </Szolgáltatás>
    <Szolgáltatás D_ID="2003" SZ_ID="3003" Útlevél="87654367">
        <típus>Bőrönd mérés</típus>
```

```
    </Szolgáltatás>
    <Dolgozó D_ID="2001" Reptér_ID="1">
        <név>Balla Sándor</név>
        <munkakör>Biztonságiőr</munkakör>
        <lakcím>Magyarország, 3530 Miskolc Arany János u. 32</lakcím>
        <telefonszám>+36204567534</telefonszám>
        <fizetés>450000</fizetés>
        <pozício>Biztonsági vezető</pozício>
    </Dolgozó>
    <Dolgozó D_ID="2002" Reptér_ID="3">
        <név>Kobe Briant</név>
        <munkakör>Felszolgáló</munkakör>
        <lakcím>USA, CA 90212 Beverly Hills Rodeo Drive 25</lakcím>
        <telefonszám>+36203641243</telefonszám>
        <fizetés>750000</fizetés>
        <pozício>Légi forgalmi irányító</pozício>
    </Dolgozó>
    <Dolgozó D_ID="2003" Reptér_ID="2">
        <név>Charles Hamilton</név>
        <munkakör>bőröndmérő</munkakör>
        <lakcím>Nagy-Britannia, SW1A 2AD London Downing Street 76</lakcím>
        <telefonszám>+36301237659</telefonszám>
        <telefonszám>+36708646856</telefonszám>
        <telefonszám>+36207363457</telefonszám>
        <fizetés>350000</fizetés>
        <pozício>Utasellátó</pozício>
    </Dolgozó>
    <Reptér Reptér_ID="1">
        <elhelyezkedés>
            <ország>Magyarország</ország>
            <város>Budapest</város>
            <irányítószám>1185</irányítószám>
        </elhelyezkedés>
    </Reptér>
    <Reptér Reptér_ID="2">
        <elhelyezkedés>
            <ország>Egyesült Királyság</ország>
            <város>London</város>
            <irányítószám>TW62GA</irányítószám>
        </elhelyezkedés>
    </Reptér>
    <Reptér Reptér_ID="3">
        <elhelyezkedés>
            <ország>Egyesült Államok</ország>
            <város>Atlanta</város>
            <irányítószám>30337</irányítószám>
        </elhelyezkedés>
    </Reptér>
    <Légitársaság Légitársaság_ID="11" Reptér_ID="1">
        <légitársaságnév>Lufthansa</légitársaságnév>
    </Légitársaság>
    <Légitársaság Légitársaság_ID="12" Reptér_ID="2">
        <légitársaságnév>Qatar Airways</légitársaságnév>
    </Légitársaság>
    <Légitársaság Légitársaság_ID="13" Reptér_ID="3">
        <légitársaságnév>Emirates</légitársaságnév>
    </Légitársaság>
</Reptér_JJL4WE>
```

Java kód:

```
package hu.domparse.jjl4we;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Comment;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DOMReadJjl4we {

    public static void main(String[] args) {
        try {
            // Létrehoz egy új XML fájlt
            File newXmlFile = new File("XMLReadjjl4we.xml");
            StreamResult newXmlStream = new StreamResult(newXmlFile);

            // Olvassa be az eredeti XML fájlt és távolítsa el az üres
szövegeket
            Document document = parseXML("XMLJJL4WE.xml");

            //Kiírás fileba
            writeDocument(document, newXmlStream);

            // Kiírja a formázott XML-t a konzolra
            System.out.println("Formázott xml:\n\n" + formatXML(document));
        } catch (IOException | SAXException | ParserConfigurationException |
TransformerException e) {
            e.printStackTrace();
        }
    }

    // XML fájl beolvasása
```

```
    public static Document parseXML(String fileName) throws SAXException,
IOException, ParserConfigurationException {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        // Dokumentum builder létrehozása a DOM objektumok létrehozásához
        DocumentBuilder builder = factory.newDocumentBuilder();
        // XML fájl beolvasása és DOM dokumentummá alakítása
        Document document = builder.parse(new File(fileName));
        // Üres szövegek eltávolítása
        cleanDocument(document.getDocumentElement());
        return document;
    }

    // Rekurzív függvény az üres szövegek eltávolítására
    private static void cleanDocument(Node root) {
        // A gyökérelem összes gyermek elemének lekérése
        NodeList nodes = root.getChildNodes();
        // Az üres szövegek eltávolítandó listájának inicializálása
        List<Node> toDelete = new ArrayList<>();
        // Az összes gyermek elem ellenőrzése
        for (int i = 0; i < nodes.getLength(); i++) {
            // Ellenőrzi, hogy a jelenlegi elem szöveges (TEXT_NODE) típusú és
üres-e
            if (nodes.item(i).getNodeType() == Node.TEXT_NODE &&
nodes.item(i).getTextContent().strip().isEmpty()) {
                //Ha üres listához adja
                toDelete.add(nodes.item(i));
            } else {
                cleanDocument(nodes.item(i));
            }
        }
        // Az üres szövegeket tartalmazó elemek eltávolítása a DOM
dokumentumból
        for (Node node : toDelete) {
            root.removeChild(node);
        }
    }

    // XML fájl írása
    public static void writeDocument(Document document, StreamResult output)
throws TransformerException {
        // TransformerFactory létrehozása
        TransformerFactory transformerFactory =
TransformerFactory.newInstance();
        // Transformer létrehozása a formázási beállításokkal
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        // DOM forrás létrehozása a dokumentumból
        DOMSource source = new DOMSource(document);
```

```
        transformer.transform(source, output);
    }

    // Formázott XML szöveg létrehozása
    public static String formatXML(Document document) {
        return "<?xml version=\"" + document.getXmlVersion() + "\"
encoding=\"" + document.getXmlEncoding() + "\" ?>\n" +
                formatElement(document.getDocumentElement(), 0);
    }

    // Rekurzív függvény az XML elemek formázására
    public static String formatElement(Node node, int indent) {
        // Ellenőrzés, hogy a Node objektum egy ELEMENT_NODE típusú-e
        if (node.getNodeType() != Node.ELEMENT_NODE) {
            return "";
        }
        // StringBuilder létrehozása a formázott XML szöveg gyűjtésére
        StringBuilder output = new StringBuilder();
        // Nyitó címke (tag) hozzáadása a StringBuilder-hez
        output.append(getIndentation(indent)).append("<").append(((Element)
node).getTagName());
        // Ha a Node objektumnak vannak attribútumai, azok hozzáadása a
StringBuilder-hez
        if (node.hasAttributes()) {
            for (int i = 0; i < node.getAttributes().getLength(); i++) {
                Node attribute = node.getAttributes().item(i);
                output.append("
").append(attribute.getNodeName()).append("=\"").append(attribute.getNodeValue
()).append("\"");
            }
        }
        // A Node objektum gyerekeinek lekérése
        NodeList children = node.getChildNodes();
        // Ha csak egy szöveges tartalom van, azt egy sorban megjelenítjük
        if (children.getLength() == 1 && children.item(0).getNodeType() ==
Node.TEXT_NODE) {
            output.append(">").append(children.item(0).getTextContent().trim()
).append("</").append(((Element) node).getTagName()).append(">\n");
        } else {
            // Nyitó címke (tag) befejezése és újsor karakter hozzáadása
            output.append(">\n");
            // Gyerekek formázása rekurzívan a megfelelő indentációval
            for (int i = 0; i < children.getLength(); i++) {
                output.append(formatElement(children.item(i), indent + 1));
            }
            // Befejező címke (tag) hozzáadása a StringBuilder-hez, újsor
karakterrel és indentációval
            output.append(getIndentation(indent)).append("</").append(((Elemen
t) node).getTagName()).append(">\n");
```

```
        }
        // A StringBuilder tartalmának visszaadása formázott XML szövegként
        return output.toString();
    }

    // Üres szóközök generálása az indentation szám alapján
    private static String getIndentation(int indent) {
        StringBuilder indentation = new StringBuilder();
        for (int i = 0; i < indent; i++) {
            indentation.append("    "); // 4 spaces for each level of
indentation
        }
        return indentation.toString();
    }
}
```

5b.) DOM adatmódosítás

A DomModifyJjl4we osztály meghatározott módosításokat hajt végre a beolvasott DOM fában, valamint az eredményül kapott módosult DOM fát kiírja konzolra és a fileba a futás végén.

Módosítások:

```
1. Névmódosítás ID alapon
2. Célmódosítás ID alapon
3. Minden dolgozónak emeljük a fizetését 10%-al akinek a megadott összeg
alatti a fizetése
4. Adjunk hozzá egy új telefonszámot az adott IDhez a dolgozókon belül
5.Születési év módosítás Név lakcím és nem alapján
```

```
package hu.domparse.jjl4we;

import java.io.File;
import java.io.IOException;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
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;
import org.xml.sax.SAXException;

public class DOMModifyJjl4we {
    public static void main(String[] args) throws TransformerException {
```

```
        try {
            Document document = DOMReadJjl4we.parseXML("XMLJJL4WE.xml");
            File newXmlFile = new File("XMLModifyjjl4we.xml");
            StreamResult newXmlStream = new StreamResult(newXmlFile);

            // 1. Névmódosítás ID alapon
            modifyUtasNev(document, "98765432", "Nagy Gergő");
            // 2. Célmódosítás ID alapon
            modifyCélállomás(document, "112", "Dallas");
            // 3. Minden dolgozónak emeljük a fizetését 10%-al akinek a
megadott összeg
            // alatti a fizetése
            modifyFizetés(document, 500000);
            // 4. Adjunk hozzá egy új telefonszámot az adott IDhez a
dolgozókon belül
            addTelefonszám(document, "2001", "+36202233567");
            // 5.Szültési év módosítás Név lakcím és nem alapján
            modifySzuletesiEv(document, "Nagy Gergő", "Magyarország, 3527
Miskolc József Attila u. 12", "Férfi", 2000);

            System.out.println("A módosított dokumentum:");
            System.out.println(DOMReadJjl4we.formatXML(document));
            writeDocument(document, newXmlStream);
            // modifyOsztalyEsSzekszam(document, "Heathrow", "Egyesült
Királyság",
            // "2023/11/18 20:00", "1", "70");
        } catch (IOException | SAXException | ParserConfigurationException e)
{
            System.out.println(e.getMessage());
        }
    }

    public static void modifyUtasNev(Document document, String utlevel, String
ujNev) {
        NodeList utasokList = document.getElementsByTagName("Utasok");

        for (int i = 0; i < utasokList.getLength(); i++) {
            Node utasNode = utasokList.item(i);

            if (utasNode.getNodeType() == Node.ELEMENT_NODE) {
                Element utasElement = (Element) utasNode;
                String utlevelAttr = utasElement.getAttribute("Útlevél");

                // Ellenőrizzük az útlevél azonosítót
                if (utlevel.equals(utlevelAttr)) {
                    // Módosítsuk az utas nevét
                    NodeList nevList =
utasElement.getElementsByTagName("név");
```

```
                    Element nevElement = (Element) nevList.item(0);
                    nevElement.setTextContent(ujNev);

                    System.out.println("Az utas neve módosítva: " + ujNev);
                    return; // Kilépünk a ciklusból, mivel megtaláltuk és
módosítottuk az utast
                }
            }
        }

        // Ha nem találtuk meg az utast
        System.out.println("Nem található ilyen útlevél azonosítóval
rendelkező utas: " + utlevel);
    }

    public static void modifyCélállomás(Document document, String járatszám,
String újCélállomás) {
        NodeList járatList = document.getElementsByTagName("Járat");

        for (int i = 0; i < járatList.getLength(); i++) {
            Node járatNode = járatList.item(i);

            if (járatNode.getNodeType() == Node.ELEMENT_NODE) {
                Element járatElement = (Element) járatNode;
                String járatszámAttr = járatElement.getAttribute("Járatszám");

                // Ellenőrizzük a járatszám azonosítót
                if (járatszám.equals(járatszámAttr)) {
                    // Módosítsuk a célállomást
                    NodeList célállomásList =
járatElement.getElementsByTagName("célállomás");
                    Element célállomásElement = (Element)
célállomásList.item(0);
                    célállomásElement.setTextContent(újCélállomás);

                    System.out.println("A célállomás módosítva: " +
újCélállomás);
                    return; // Kilépünk a ciklusból, mivel megtaláltuk és
módosítottuk a járatot
                }
            }
        }

        // Ha nem találtuk meg a járatot
        System.out.println("Nem található ilyen járatszám azonosítóval
rendelkező járat: " + járatszám);
    }
```

```
    public static void modifyFizetés(Document document, double
referenciaFizetés) {
        NodeList dolgozóList = document.getElementsByTagName("Dolgozó");

        for (int i = 0; i < dolgozóList.getLength(); i++) {
            Node dolgozóNode = dolgozóList.item(i);

            if (dolgozóNode.getNodeType() == Node.ELEMENT_NODE) {
                Element dolgozóElement = (Element) dolgozóNode;
                String fizetésString =
dolgozóElement.getElementsByTagName("fizetés").item(0).getTextContent();
                double fizetés = Double.parseDouble(fizetésString);

                // Ellenőrizzük a fizetést
                if (fizetés < referenciaFizetés) {
                    // Módosítsuk a fizetést és kerekítsük az egész részre
                    double újFizetés = Math.round(fizetés * 1.1);
                    Element fizetésElement = (Element)
dolgozóElement.getElementsByTagName("fizetés").item(0);
                    fizetésElement.setTextContent(String.valueOf((int)
újFizetés));

                    System.out.println("A fizetés módosítva a dolgozónál: "
                            +
dolgozóElement.getElementsByTagName("név").item(0).getTextContent() + ", Új
fizetés: "
                            + (int) újFizetés);
                }
            }
        }
    }

    public static void writeDocument(Document document, StreamResult output)
throws TransformerException {
        TransformerFactory transformerFactory =
TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");

        DOMSource source = new DOMSource(document);
        transformer.transform(source, output);
    }

    public static void addTelefonszám(Document document, String dolgozoId,
String ujTelefonszam) {
        NodeList dolgozoList = document.getElementsByTagName("Dolgozó");

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

```
            Node dolgozoNode = dolgozoList.item(i);

            if (dolgozoNode.getNodeType() == Node.ELEMENT_NODE) {
                Element dolgozoElement = (Element) dolgozoNode;
                String dolgozoIdAttr = dolgozoElement.getAttribute("D_ID");

                // Ellenőrizzük a Dolgozó-ID-t
                if (dolgozoId.equals(dolgozoIdAttr)) {
                    // Hozzáadjuk az új telefonszámot közvetlenül a meglévő
telefonszámok után
                    NodeList telefonszamList =
dolgozoElement.getElementsByTagName("telefonszám");
                    Element lastTelefonszamElement = (Element)
telefonszamList.item(telefonszamList.getLength() - 1);
                    Element telefonszamElement =
document.createElement("telefonszám");
                    telefonszamElement.setTextContent(ujTelefonszam);
                    dolgozoElement.insertBefore(telefonszamElement,
lastTelefonszamElement.getNextSibling());

                    System.out.println(
                            "Új telefonszám hozzáadva a dolgozónak (D_ID: " +
dolgozoId + "): " + ujTelefonszam);
                    return; // Kilépünk a ciklusból, mivel megtaláltuk és
hozzáadtuk a telefonszámot
                }
            }
        }

        // Ha nem találtuk meg a dolgozót
        System.out.println("Nem található ilyen Dolgozó-ID-vel rendelkező
dolgozó: " + dolgozoId);
    }

    /*public static void modifyOsztalyEsSzekszam(Document document, String
repterNev, String orszag, String indulasiIdo,
            String ujOsztaly, String ujSzekszam) {
        NodeList jegyList = document.getElementsByTagName("Jegy");

        for (int i = 0; i < jegyList.getLength(); i++) {
            Node jegyNode = jegyList.item(i);

            if (jegyNode.getNodeType() == Node.ELEMENT_NODE) {
                Element jegyElement = (Element) jegyNode;
                Element uticelElement = (Element)
jegyElement.getElementsByTagName("uticél").item(0);
                String jegyRepterNev =
uticelElement.getElementsByTagName("reptérnév").item(0).getTextContent();
```

```
                String jegyOrszag =
uticelElement.getElementsByTagName("ország").item(0).getTextContent();
                String jegyIndulasiIdo =
jegyElement.getElementsByTagName("indulási_idő").item(0).getTextContent();

                // Ellenőrizzük a feltételeket
                if (jegyRepterNev.equals(repterNev) &&
jegyOrszag.equals(orszag)
                        && jegyIndulasiIdo.equals(indulasiIdo)) {
                    // Módosítjuk az osztályt és a székszámot
                    Element helyElement = (Element)
jegyElement.getElementsByTagName("hely").item(0);
                    Element osztalyElement = (Element)
helyElement.getElementsByTagName("osztály").item(0);
                    Element szekszamElement = (Element)
helyElement.getElementsByTagName("székszám").item(0);

                    osztalyElement.setTextContent(ujOsztaly);
                    szekszamElement.setTextContent(ujSzekszam);

                    System.out.println(
                            "Osztály és székszám módosítva a jegynél: " +
jegyElement.getAttribute("Jegysorszám"));
                    return; // Kilépünk a ciklusból, mivel megtaláltuk és
módosítottuk a Jegyet
                }
            }
        }
    }
*/
    public static void modifySzuletesiEv(Document document, String nev, String
lakcim, String nem, int ujSzuletesiEv) {
        NodeList utasokList = document.getElementsByTagName("Utasok");

        for (int i = 0; i < utasokList.getLength(); i++) {
            Node utasokNode = utasokList.item(i);

            if (utasokNode.getNodeType() == Node.ELEMENT_NODE) {
                Element utasokElement = (Element) utasokNode;
                String utasNev =
utasokElement.getElementsByTagName("név").item(0).getTextContent();
                String utasLakcim =
utasokElement.getElementsByTagName("lakcím").item(0).getTextContent();
                String utasNem =
utasokElement.getElementsByTagName("nem").item(0).getTextContent();

                // Ellenőrizzük a feltételeket
                if (utasNev.equals(nev) && utasLakcim.equals(lakcim) &&
utasNem.equals(nem)) {
```

```
                    // Módosítjuk a születési évet
                    Element szuletesiEvElement = (Element)
utasokElement.getElementsByTagName("születésiév").item(0);
                    szuletesiEvElement.setTextContent(String.valueOf(ujSzulete
siEv));

                    System.out.println("Születési év módosítva az utasnak
(Név: " + nev + ", Lakcím: " + lakcim
                            + ", Nem: " + nem + "): " + ujSzuletesiEv);
                    return; // Kilépünk a ciklusból, mivel megtaláltuk és
módosítottuk az Utast
                }
            }
        }
    }
}
```

5c.) DOM adatlekérdezés

A DOMQueryJjl4we osztály meghatározott lekérdezéseket hajt végre a beolvasott XML dokumentumra vonatkozóan, melyek eredményeit a konzolra írja ki.

lekérdezések:

// 1. Összes utas név

// 2. Összes járat indulási idővel

// 3.Az összes dolgozó kevesebb mint 500000 fizetéssel

// 4 Átlagos fizetés

// 5 legtöbbet kereső dolgozó

```
package hu.domparse.jjl4we;

import java.io.IOException;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DOMQueryJjl4we {
    public static void main(String[] args) {
        try {
            Document document = DOMReadJjl4we.parseXML("XMLJJL4WE.xml");
```

```
            // 1. Összes utas név
            String osszesUtasNev = getAllPassangerName(document);
            System.out.println("Az összes utas neve: ");
            System.out.println(osszesUtasNev);

            // 2. Összes járat indulási idővel
            String jaratIndulasiIdovel = getAllPlanes(document);
            System.out.println("Az összes járat indulási idővel: ");
            System.out.println(jaratIndulasiIdovel);

            // 3.Az összes dolgozó kevesebb mint 500000 fizetéssel
            String dolgozofizetes = getAllEployesWithSalary(document, 449000);
            System.out.println("Az összes dolgozó kevesebb mint 500000
fizetéssel");
            System.out.println(dolgozofizetes);
            // 4 Átlagos fizetés
            String atlagDolgozofizetes = getAverageSalary(document);
            System.out.println(atlagDolgozofizetes);

            // 5 legtöbbet kereső dolgozó
            printHighestEarningEmployee(document);

        } catch (IOException | SAXException | ParserConfigurationException e)
{
            System.out.println(e.getMessage());
        }
    }

    public static String getAllPassangerName(Document document) {
        StringBuilder result = new StringBuilder();

        // Lekérdezés az Utasok elemekre
        NodeList utasokList = document.getElementsByTagName("Utasok");

        for (int i = 0; i < utasokList.getLength(); i++) {
            Element utasokElement = (Element) utasokList.item(i);

            // Név lekérdezése az Utasok elem alól
            String nev =
utasokElement.getElementsByTagName("név").item(0).getTextContent();

            // Eredményhez hozzáadás
            result.append(nev).append("\n");
        }

        return result.toString();
    }
```

```
    public static String getAllPlanes(Document document) {
        StringBuilder result = new StringBuilder();

        // Lekérdezés a Jarat elemekre
        NodeList jaratokList = document.getElementsByTagName("Járat");

        for (int i = 0; i < jaratokList.getLength(); i++) {
            Element jaratElement = (Element) jaratokList.item(i);

            // Járat adatainak lekérdezése
            String celallomas =
jaratElement.getElementsByTagName("célállomás").item(0).getTextContent();
            String indulasiIdo =
jaratElement.getElementsByTagName("indulási_idő").item(0).getTextContent();

            // Eredményhez hozzáadás
            result.append("Célállomás: ").append(celallomas).append(",
Indulási idő: ").append(indulasiIdo)
                    .append("\n");
        }

        return result.toString();
    }

    public static String getAllEployesWithSalary(Document document, int
minSalary) {
        StringBuilder result = new StringBuilder();


        // Lekérdezés a Dolgozó elemekre
        NodeList dolgozokList = document.getElementsByTagName("Dolgozó");

        for (int i = 0; i < dolgozokList.getLength(); i++) {
            Element dolgozoElement = (Element) dolgozokList.item(i);

            // Fizetés lekérdezése a Dolgozo elem alól
            int fizetes =
Integer.parseInt(dolgozoElement.getElementsByTagName("fizetés").item(0).getTex
tContent());

            // Ellenőrzés, hogy a fizetés kisebb-e a megadott értéknél
            if (fizetes < minSalary) {
                // Név lekérdezése a Dolgozo elem alól
                String nev =
dolgozoElement.getElementsByTagName("név").item(0).getTextContent();

                // Eredményhez hozzáadás
                result.append(nev).append("\n");
            }
```

```
        }

        return result.toString();
    }

    public static String getAverageSalary(Document document) {
        NodeList dolgozokList = document.getElementsByTagName("Dolgozó");

        int totalSalary = 0;
        int numberOfEmployees = dolgozokList.getLength();

        for (int i = 0; i < numberOfEmployees; i++) {
            Element dolgozoElement = (Element) dolgozokList.item(i);
            int fizetes =
Integer.parseInt(dolgozoElement.getElementsByTagName("fizetés").item(0).getTex
tContent());
            totalSalary += fizetes;
        }

        if (numberOfEmployees > 0) {
            double averageSalary = (double) totalSalary / numberOfEmployees;
            return "Az átlagfizetés a dolgozók között: \n" + averageSalary;
        } else {
            return "Nincs elérhető dolgozói információ.";
        }
    }

    public static void printHighestEarningEmployee(Document document) {
        NodeList dolgozokList = document.getElementsByTagName("Dolgozó");

        int maxSalary = 0;
        String highestEarningEmployeeName = "";

        for (int i = 0; i < dolgozokList.getLength(); i++) {
            Element dolgozoElement = (Element) dolgozokList.item(i);
            int fizetes =
Integer.parseInt(dolgozoElement.getElementsByTagName("fizetés").item(0).getTex
tContent());

            if (fizetes > maxSalary) {
                maxSalary = fizetes;
                highestEarningEmployeeName =
dolgozoElement.getElementsByTagName("név").item(0).getTextContent();
            }
        }

        if (!highestEarningEmployeeName.isEmpty()) {
            System.out.println("\nA legtöbbet kereső dolgozó neve: \n" +
highestEarningEmployeeName);
```

```
        } else {
            System.out.println("Nincs elérhető dolgozói információ.");
        }
    }

}
```

5d.) DOM adatírás

A DomWriteJjl4we osztály createAirport() metódusa felépíti a Reptér_JJL4WE tartozó DOM fát, melyet aztán a DOMReadJjl4we osztályt felhasználva kiír a konzolra és XML fájlba is, melynek neve XMLJjl4we1.xml. A különböző típusú létrehozására paraméterezhető metódusokat írtam.

```
package hu.domparse.jjl4we;

import java.io.File;
import java.text.ParseException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.DOMException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class DOMWriteJjl4we {
    public static void main(String[] args) throws ParseException {
        try {
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.newDocument();

            createAirport(document);


            // Kiírás konzolra és fájlba a DOMReadJjl4we osztályt felhasználva
            File newXmlFile = new File("XMLJjl4we1.xml");
            StreamResult xmlToWrite = new StreamResult(newXmlFile);
            DOMReadJjl4we.writeDocument(document, xmlToWrite);
            StreamResult console = new StreamResult(System.out);
            System.out.println("A felépített dokumentum:\n");
            DOMReadJjl4we.writeDocument(document, console);

        } catch (ParserConfigurationException | TransformerException |
DOMException e) {
```

```
            e.printStackTrace();
        }
}
    // Járat elem létrehozása és hozzáadása a DOM fához
    public static Element createJárat(Document document, String járatszám,
String légitársaságID,
            String célállomás, String indulásiIdő, String érkezésiIdő, String
státusz) {
        Element járatElement = document.createElement("Járat");
        járatElement.setAttribute("Járatszám", járatszám);
        járatElement.setAttribute("Légitársaság_ID", légitársaságID);

        createElementWithTextContent(document, járatElement, "célállomás",
célállomás);
        createElementWithTextContent(document, járatElement, "indulási_idő",
indulásiIdő);
        createElementWithTextContent(document, járatElement, "érkezési_idő",
érkezésiIdő);
        createElementWithTextContent(document, járatElement, "státusz",
státusz);

        return járatElement;
    }

    // Jegy elem létrehozása és hozzáadása a DOM fához
    public static Element createJegy(Document document, String jegySorszám,
String ár,
            String légitársaság, String reptérNév, String ország, String
indulásiIdő, String érkezésiIdő,
            String osztály, String székszám) {
        Element jegyElement = document.createElement("Jegy");
        jegyElement.setAttribute("Jegysorszám", jegySorszám);

        createElementWithTextContent(document, jegyElement, "ár", ár);
        createElementWithTextContent(document, jegyElement, "légitársaság",
légitársaság);

        Element uticélElement = document.createElement("uticél");
        createElementWithTextContent(document, uticélElement, "reptérnév",
reptérNév);
        createElementWithTextContent(document, uticélElement, "ország",
ország);
        jegyElement.appendChild(uticélElement);

        createElementWithTextContent(document, jegyElement, "indulási_idő",
indulásiIdő);
        createElementWithTextContent(document, jegyElement, "érkezési_idő",
érkezésiIdő);
```

```
        Element helyElement = document.createElement("hely");
        createElementWithTextContent(document, helyElement, "osztály",
osztály);
        createElementWithTextContent(document, helyElement, "székszám",
székszám);
        jegyElement.appendChild(helyElement);

        return jegyElement;
    }

    // Utasok elem létrehozása és hozzáadása a DOM fához
    public static Element createUtasok(Document document, String útlevél,
String jegySorszám,
            String járatszám, String név, String születésiÉv, String nem,
String lakcím,
            String... telefonszámok) {
        Element utasokElement = document.createElement("Utasok");
        utasokElement.setAttribute("Útlevél", útlevél);
        utasokElement.setAttribute("Jegysorszám", jegySorszám);
        utasokElement.setAttribute("Járatszám", járatszám);

        createElementWithTextContent(document, utasokElement, "név", név);
        createElementWithTextContent(document, utasokElement, "születésiév",
születésiÉv);
        createElementWithTextContent(document, utasokElement, "nem", nem);
        createElementWithTextContent(document, utasokElement, "lakcím",
lakcím);

        for (String telefonszám : telefonszámok) {
            createElementWithTextContent(document, utasokElement,
"telefonszám", telefonszám);
        }

        return utasokElement;
    }

    // Szolgáltatás elem létrehozása és hozzáadása a DOM fához
    public static Element createSzolgáltatás(Document document, String
szolgáltatásID,
            String dolgozóID, String útlevél, String típus) {
        Element szolgáltatásElement = document.createElement("Szolgáltatás");
        szolgáltatásElement.setAttribute("SZ_ID", szolgáltatásID);
        szolgáltatásElement.setAttribute("D_ID", dolgozóID);
        szolgáltatásElement.setAttribute("Útlevél", útlevél);

        createElementWithTextContent(document, szolgáltatásElement, "típus",
típus);

        return szolgáltatásElement;
```

```
    }

    // Dolgozó elem létrehozása és hozzáadása a DOM fához
    public static Element createDolgozó(Document document, String dolgozóID,
String reptérID,
            String név, String munkakör, String lakcím, String fizetés, String
pozíció, String... telefonszámok) {
        Element dolgozóElement = document.createElement("Dolgozó");
        dolgozóElement.setAttribute("D_ID", dolgozóID);
        dolgozóElement.setAttribute("Reptér_ID", reptérID);

        createElementWithTextContent(document, dolgozóElement, "név", név);
        createElementWithTextContent(document, dolgozóElement, "munkakör",
munkakör);
        createElementWithTextContent(document, dolgozóElement, "lakcím",
lakcím);

        for (String telefonszám : telefonszámok) {
            createElementWithTextContent(document, dolgozóElement,
"telefonszám", telefonszám);
        }

        createElementWithTextContent(document, dolgozóElement, "fizetés",
fizetés);
        createElementWithTextContent(document, dolgozóElement, "pozíció",
pozíció);

        return dolgozóElement;
    }

    // Reptér elem létrehozása és hozzáadása a DOM fához
    public static Element createReptér(Document document, String reptérID,
String ország,
            String város, String irányítószám) {
        Element reptérElement = document.createElement("Reptér");
        reptérElement.setAttribute("Reptér_ID", reptérID);

        Element elhelyezkedésElement =
document.createElement("elhelyezkedés");
        createElementWithTextContent(document, elhelyezkedésElement, "ország",
ország);
        createElementWithTextContent(document, elhelyezkedésElement, "város",
város);
        createElementWithTextContent(document, elhelyezkedésElement,
"irányítószám", irányítószám);
        reptérElement.appendChild(elhelyezkedésElement);

        return reptérElement;
    }
```

```
    // Légitársaság elem létrehozása és hozzáadása a DOM fához
    public static Element createLégitársaság(Document document, String
légitársaságID,
            String reptérID, String légitársaságNév) {
        Element légitársaságElement = document.createElement("Légitársaság");
        légitársaságElement.setAttribute("Légitársaság_ID", légitársaságID);
        légitársaságElement.setAttribute("Reptér_ID", reptérID);

        createElementWithTextContent(document, légitársaságElement,
"légitársaságnév", légitársaságNév);

        return légitársaságElement;
    }

    // Segédfüggvény szöveges tartalommal rendelkező elem létrehozására és
hozzáadására
    private static void createElementWithTextContent(Document document,
Element parentElement,
            String elementName, String textContent) {
        Element element = document.createElement(elementName);
        element.setTextContent(textContent);
        parentElement.appendChild(element);
    }
    private static void createAirport(Document document) throws DOMException,
ParseException {
        // Gyökérelem felvétele
        Element root = document.createElement("Reptér_JJL4WE");
        document.appendChild(root);

        // Járat felvétele
        root.appendChild(document.createComment("Járat"));
        root.appendChild(createJárat(document, "111", "12", "Budapest",
"2023/11/17 17:00", "2023/11/17 19:00", "várható"));
        root.appendChild(createJárat(document, "112", "11", "London",
"2023/11/18 20:00", "2023/11/19 01:00", "késik"));
        root.appendChild(createJárat(document, "113", "13", "Atlanta",
"2023/11/20 11:00", "2023/11/20 15:00", "várható"));
        //Jegy felvétele
        root.appendChild(document.createComment("Jegy"));
        root.appendChild(createJegy(document, "1111", "40000", "Lufthansa",
"Heathrow", "Egyesült Királyság", "2023/11/18 20:00", "2023/11/19 01:00", "1",
"31"));
        root.appendChild(createJegy(document, "1112", "20000", "Qatar
Airways", "Ferihegy", "Magyarország", "2023/11/17 17:00", "2023/11/17 19:00",
"2", "52"));
        root.appendChild(createJegy(document, "1113", "50000", "Emirates",
"Hartsfield-Jackson", "Egyesült Államok", "2023/11/20 11:00", "2023/11/20
15:00", "1", "11"));
```

```
        //Utasok felvétele
        root.appendChild(document.createComment("Utasok"));
        root.appendChild(createUtasok(document, "98765432", "1113", "113",
"Hauer Attila", "2002", "Férfi", "Magyarország, 3527 Miskolc József Attila u.
12","+36201234567","+36701234567"));
        root.appendChild(createUtasok(document, "87654367", "1112", "111",
"Olivia Thompson", "2001", "Nő", "Nagy-Britannia, SW1A 2AB London Downing
Street 10","+36209876428","+36702138757","+36705834653"));
        root.appendChild(createUtasok(document, "87575645", "1111", "112",
"Benjamin Mitchell", "1970", "Nő", "USA, CA 90212 Beverly Hills Rodeo Drive
123","+36202345965"));
        //Szolgáltatás felvétele
        root.appendChild(document.createComment("Szolgáltatás"));
        root.appendChild(createSzolgáltatás(document, "3001", "2001",
"98765432", "Információ nyújtás"));
        root.appendChild(createSzolgáltatás(document, "3002", "2002",
"87575645", "Felszolgálás"));
        root.appendChild(createSzolgáltatás(document, "3003", "2003",
"87654367", "Bőrönd mérés"));
        //Dolgozók felvétele
        root.appendChild(document.createComment("Dolgozó"));
        root.appendChild(createDolgozó(document, "2001", "1", "Balla Sándor",
"Biztonságiőr", "Magyarország, 3530 Miskolc Arany János u. 32", "450000",
"Biztonsági vezető", "+36204567534"));
        root.appendChild(createDolgozó(document, "2002", "3", "Kobe Briant",
"Felszolgáló", "USA, CA 90212 Beverly Hills Rodeo Drive 25", "750000", "Légi
forgalmi irányító", "+36203641243"));
        root.appendChild(createDolgozó(document, "2003", "2", "Charles
Hamilton", "bőröndmérő", "Nagy-Britannia, SW1A 2AD London Downing Street 76",
"350000", "Utasellátó", "+36301237659","+36708646856","+36207363457"));
        //Reptér felvétele
        root.appendChild(document.createComment("Reptér"));
        root.appendChild(createReptér(document, "1", "Magyarország",
"Budapest", "1185"));
        root.appendChild(createReptér(document, "2", "Egyesült Királyság",
"London", "TW62GA"));
        root.appendChild(createReptér(document, "3", "Egyesült Államok",
"Atlanta", "30337"));
        //Légitársaság felvétele
        root.appendChild(document.createComment("Légitársaság"));
        root.appendChild(createLégitársaság(document, "11", "1",
"Lufthansa"));
        root.appendChild(createLégitársaság(document, "12", "2", "Qatar
Airways"));
        root.appendChild(createLégitársaság(document, "13", "3", "Emirates"));
    }
}
```