JEGYZŐKÖNYV

Adatkezelés XML-ben

Féléves feladat

Légiforgalom

Készítette: **Barabás Márton**

Neptunkód: **D21YGM**

Dátum: **2024-12-09**

Tartalom

1. A feladat leírása

A modern légiforgalom a globális közlekedés egyik legdinamikusabban fejlődő ágazata, amely nap mint nap emberek millióinak életét érinti. A repülésbiztonság, az utasok kényelme és a hatékony forgalomirányítás érdekében kulcsfontosságú a pontos adatok kezelése és nyomon követése. Egy jól működő nyilvántartó rendszer nem csupán az adatgyűjtést és tárolást teszi lehetővé, hanem elősegíti a döntéshozatalt, az elemzést és a jövőbeli fejlesztéseket is.

Ez a rendszer a légiforgalom különböző aspektusaira koncentrál, legyen szó repülőgépek nyomon követéséről, járatok menetrendjének kezeléséről, a repterek tulajdonságairól vagy a személyzetről.

2. Első feladat

2.1. ER Modell

Kapitány egyed tulajdonságai:

- Személyi azonosító: PK
- Vezetéknév
- Keresztnév
- Repült őrák száma

Repülő egyed tulajdonságai:

- Hívójel: PK
- Kapitány: FK
- Gyártó
- Típus
- Szín

Járat egyed tulajdonságai:

- Járat azonosító: FK
- Indulás
- Érkezés
- Hossz

Reprér egyed tulajdonságai:

- IATA: PK
- Ország
- Város
- Légiforgalom irányítás

Kifutópálya egyed tulajdonságai:

- Kifutópálya azonosító: PK
- Fizikai paraméterek:
  - Meredekség
  - Hossz
  - Irány
  - Aszfaltozott

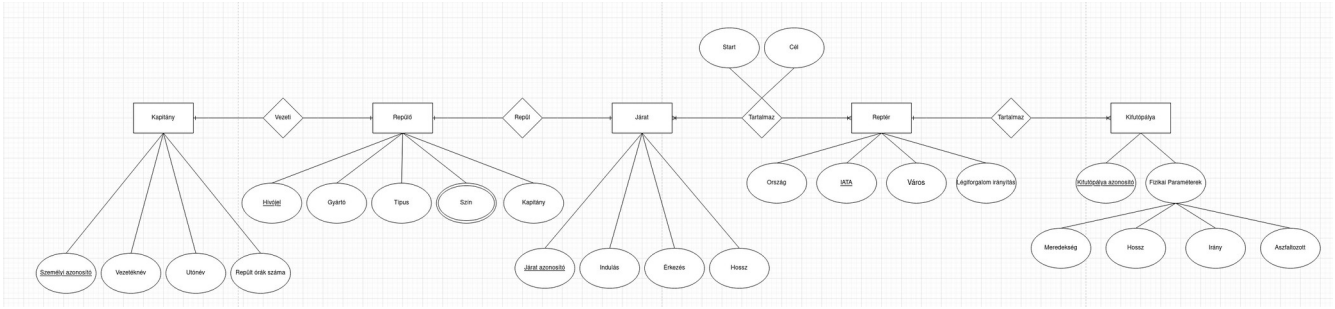

Figure 1: ER Modell

Kapitány – Repülő kapcsolat: Vezeti

Egy-Egy kapcsolat mivel egyszerre csak egy repülőt vezet egy kapitány.

Repülő – Járat kapcsolat: Repül

Egy-Egy kapcsolat hiszen egyszerre egy járatot csak egy repülő tud repülni.

Járat – Reptér kapcsolat: Tartalmaz

Több-Több kapcsolat, mivel egy reptér több járatot tartalmaz és egy járat is több repteret tartalmaz.

Reptért – Kiufutópálya kapcsolat: Tartalmaz

Egy-Több kapcsolat, mivel egy reptérnek több kifutópályája lehet de egy kifutópálya csak egy reptérhez tartozik.

2.2. XDM Modell

Az XDM modell esetén három fő jelölést tudunk elkülöníteni, elipszis amely egy elem, rombusz ami attribútum és téglalap amely tartalom. A többszöri előfordulás duplán van jelölve. A kulcsok referálása szagatott nyíllal van ábrázolva.

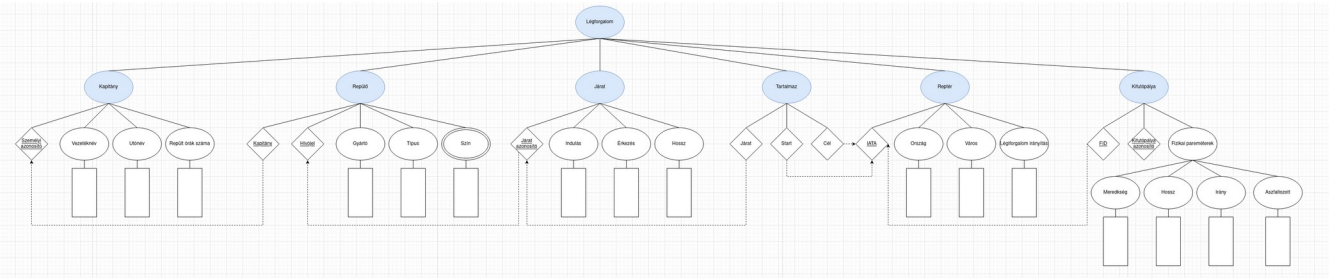

Figure 2: XDM Modell

2.3. XML Dokumentum

A fentebbi modell alapján XML dokumentumot készítettem. Minden elemből legalább három példányt csináltam.

```
<Légforgalom>

  <!-- Kapitányok -->
  <Kapitány személyi_azonosító="01">
    <Vezetéknév>Barabás</Vezetéknév>
    <Utónév>Márton</Utónév>
    <Repült>9000</Repült>
  </Kapitány>
  <Kapitány személyi_azonosító="02">
    <Vezetéknév>Tóth</Vezetéknév>
    <Utónév>Tibor</Utónév>
    <Repült>7000</Repült>
  </Kapitány>
  <Kapitány személyi_azonosító="03">
    <Vezetéknév>Példa</Vezetéknév>
    <Utónév>Béla</Utónév>
    <Repült>15000</Repült>
  </Kapitány>

  <!-- Repülőgépek -->
  <Repülő Kapitány="01" Hívójel="CPA238">
    <Gyártó>Airbus</Gyártó>
    <Típus>A400M Atlas</Típus>
    <Szín>Szürke</Szín>
  </Repülő>
  <Repülő Kapitány="02" Hívójel="RYR997">
    <Gyártó>Airbus</Gyártó>
```

```
    <Típus>A320-232</Típus>
    <Szín>Fehér Kék Sárga</Szín>
</Repülő>
<Repülő Kapitány="03" Hívójel="BCS5867">
    <Gyártó>Boeing</Gyártó>
    <Típus>737-49R</Típus>
    <Szín>Fehér</Szín>
</Repülő>

<!-- Járatok -->
<Járat id="CPA238">
    <Indulás>2024-12-07T19:00</Indulás>
    <Érkezés>2024-12-07T23:00</Érkezés>
    <Hossz>3670</Hossz>
</Járat>
<Járat id="RYR997">
    <Indulás>2024-12-07T06:30</Indulás>
    <Érkezés>2024-12-07T12:00</Érkezés>
    <Hossz>2101</Hossz>
</Járat>
<Járat id="BCS5867">
    <Indulás>2024-12-07T19:00</Indulás>
    <Érkezés>2024-12-08T01:00</Érkezés>
    <Hossz>8632</Hossz>
</Járat>

<!-- Repterek és a hozzájuk tartozó kifutópályák -->
<Reptér IATA="BUD">
    <Ország>Magyarország</Ország>
    <Város>Budapest</Város>
    <Légiforgalom>Igen</Légiforgalom>
</Reptér>
<Kifutópálya id="01" fid="BUD">
    <Paraméterek>
        <Meredekség>0</Meredekség>
        <Hossz>3707</Hossz>
        <Irány>127</Irány>
        <Aszfalt>Igen</Aszfalt>
    </Paraméterek>
</Kifutópálya>
<Kifutópálya id="02" fid="BUD">
    <Paraméterek>
        <Meredekség>0</Meredekség>
        <Hossz>3009</Hossz>
        <Irány>127</Irány>
        <Aszfalt>Igen</Aszfalt>
    </Paraméterek>
</Kifutópálya>
```

```
<Reptér IATA="LAX">
   <Ország>Amerikai Egyesült Államok</Ország>
   <Város>Los Angeles</Város>
   <Légiforgalom>Igen</Légiforgalom>
</Reptér>
<Kifutópálya id="01" fid="LAX">
   <Paraméterek>
      <Meredekség>1</Meredekség>
      <Hossz>3939</Hossz>
      <Irány>83</Irány>
      <Aszfalt>Igen</Aszfalt>
   </Paraméterek>
</Kifutópálya>
<Kifutópálya id="02" fid="LAX">
   <Paraméterek>
      <Meredekség>1</Meredekség>
      <Hossz>3383</Hossz>
      <Irány>83</Irány>
      <Aszfalt>Igen</Aszfalt>
   </Paraméterek>
</Kifutópálya>
<Kifutópálya id="03" fid="LAX">
   <Paraméterek>
      <Meredekség>1</Meredekség>
      <Hossz>3318</Hossz>
      <Irány>83</Irány>
      <Aszfalt>Igen</Aszfalt>
   </Paraméterek>
</Kifutópálya>
<Kifutópálya id="04" fid="LAX">
   <Paraméterek>
      <Meredekség>1</Meredekség>
      <Hossz>2721</Hossz>
      <Irány>83</Irány>
      <Aszfalt>Igen</Aszfalt>
   </Paraméterek>
</Kifutópálya>

<Reptér IATA="VIE">
   <Ország>Ausztria</Ország>
   <Város>Bécs</Város>
   <Légiforgalom>Igen</Légiforgalom>
</Reptér>
<Kifutópálya id="01" fid="VIE">
   <Paraméterek>
      <Meredekség>0</Meredekség>
      <Hossz>3600</Hossz>
      <Irány>159</Irány>
      <Aszfalt>Igen</Aszfalt>
```

```
    </Paraméterek>
  </Kifutópálya>
  <Kifutópálya id="02" fid="VIE">
    <Paraméterek>
      <Meredekség>0</Meredekség>
      <Hossz>3500</Hossz>
      <Irány>111</Irány>
      <Aszfalt>Igen</Aszfalt>
    </Paraméterek>
  </Kifutópálya>

  <!-- A repterek és járatok közti kapcsolat -->
  <Tartalmaz Járat="CPA238" Start="BUD" Cél="LAX"/>
  <Tartalmaz Járat="RYR997" Start="LAX" Cél="VIE"/>
  <Tartalmaz Járat="BCS5867" Start="VIE" Cél="BUD"/>

</Légforgalom>
```

2.4. XML Schema

A validálás érdekében létrehoztam egy schema-t. Elsőnek definiáltam a gyökér elemet és az elemeit. Utána a kulcsok következnek. Következő lépésben elkészítettem három darab hasznos egyszerű típust. Az elemeknek saját komplex típusokat készítettem.

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

  <!-- Gyökér elem -->
  <xs:element name="Légforgalom">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Kapitány" maxOccurs="unbounded" type="KapitányTípus"/>
        <xs:element name="Repülő" maxOccurs="unbounded" type="RepülőTípus"/>
        <xs:element name="Járat" maxOccurs="unbounded" type="JáratTípus"/>
        <xs:element name="Reptér" maxOccurs="unbounded" type="ReptérTípus"/>
        <xs:element name="Kifutópálya" maxOccurs="unbounded" type="KifutópályaTípus"/>
        <xs:element name="Tartalmaz" maxOccurs="unbounded" type="TartalmazTípus"/>
      </xs:sequence>
    </xs:complexType>

    <!-- Kulcsok -->
    <xs:key name="KapitányId">
      <xs:selector xpath="Kapitány"/>
      <xs:field xpath="@személyi_azonosító"/>
    </xs:key>
    <xs:keyref name="RepülőKapitány" refer="KapitányId">
      <xs:selector xpath="Repülő"/>
      <xs:field xpath="@Kapitány"/>
    </xs:keyref>
```

```
    <xs:key name="Hivójel">
      <xs:selector xpath="Repülő"/>
      <xs:field xpath="@Hívójel"/>
    </xs:key>
    <xs:keyref name="JáratRepülő" refer="Hívójel">
      <xs:selector xpath="Járat"/>
      <xs:field xpath="@id"/>
    </xs:keyref>

    <xs:key name="JáratId">
      <xs:selector xpath="Járat"/>
      <xs:field xpath="@id"/>
    </xs:key>
    <xs:keyref name="TartalmazJárat" refer="JáratId">
      <xs:selector xpath="Tartalmaz"/>
      <xs:field xpath="@Járat"/>
    </xs:keyref>

    <xs:key name="IATA">
      <xs:selector xpath="Reptér"/>
      <xs:field xpath="@IATA"/>
    </xs:key>
    <xs:keyref name="TartalmazStart" refer="IATA">
      <xs:selector xpath="Tartalmaz"/>
      <xs:field xpath="@Start"/>
    </xs:keyref>
    <xs:keyref name="TartalmazCél" refer="IATA">
      <xs:selector xpath="Tartalmaz"/>
      <xs:field xpath="@Cél"/>
    </xs:keyref>
    <xs:keyref name="KifutópályaFid" refer="IATA">
      <xs:selector xpath="Kifutópálya"/>
      <xs:field xpath="@fid"/>
    </xs:keyref>

  </xs:element>

  <!-- Egyszerű típusok -->
  <xs:simpleType name="IgenNemType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Igen"/>
      <xs:enumeration value="Nem"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="DátumTípus">
    <xs:restriction base="xs:string">
      <xs:pattern value="\d{4}-\d{2}-\d{2}T\d{2}:\d{2}"/>
```

```
    </xs:restriction>
  </xs:simpleType>

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

  <!-- Kapitány Típus -->
  <xs:complexType name="KapitányTípus">
    <xs:sequence>
      <xs:element name="Vezetéknév" type="xs:string"/>
      <xs:element name="Utónév" type="xs:string"/>
      <xs:element name="Repült" type="xs:integer"/>
    </xs:sequence>
    <xs:attribute name="személyi_azonosító" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- Repülő Típus -->
  <xs:complexType name="RepülőTípus">
    <xs:sequence>
      <xs:element name="Gyártó" type="xs:string"/>
      <xs:element name="Típus" type="xs:string"/>
      <xs:element name="Szín" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="Kapitány" type="xs:string" use="required"/>
    <xs:attribute name="Hívójel" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- Járat Típus -->
  <xs:complexType name="JáratTípus">
    <xs:sequence>
      <xs:element name="Indulás" type="DátumTípus"/>
      <xs:element name="Érkezés" type="DátumTípus"/>
      <xs:element name="Hossz" type="xs:integer"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- Reptér Típus -->
  <xs:complexType name="ReptérTípus">
    <xs:sequence>
      <xs:element name="Ország" type="xs:string"/>
      <xs:element name="Város" type="xs:string"/>
      <xs:element name="Légiforgalom" type="IgenNemType"/>
    </xs:sequence>
    <xs:attribute name="IATA" type="IATATípus" use="required"/>
  </xs:complexType>
```

```
  <!-- Kifutópálya Típus -->
  <xs:complexType name="KifutópályaTípus">
    <xs:sequence>
      <xs:element name="Paraméterek" type="ParaméterekTípus"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
    <xs:attribute name="fid" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- Paraméterek Típus -->
  <xs:complexType name="ParaméterekTípus">
    <xs:sequence>
      <xs:element name="Meredekség" type="xs:integer"/>
      <xs:element name="Hossz" type="xs:integer"/>
      <xs:element name="Irány" type="xs:integer"/>
      <xs:element name="Aszfalt" type="IgenNemType"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Tartalmaz Típus -->
  <xs:complexType name="TartalmazTípus">
    <xs:attribute name="Járat" type="xs:string" use="required"/>
    <xs:attribute name="Start" type="xs:string" use="required"/>
    <xs:attribute name="Cél" type="xs:string" use="required"/>
  </xs:complexType>

</xs:schema>
```

3. Második feladat

3.1. Adatolvasás

A program legelőször beolvassa az XML fájlt, amit utána normalizál. Kiválasztja a gyökér elemet majd egy segéd függvény segítségével kiírja a konzolra a tartalmat. A segéd függvény elemenként végig megy a dokumentumon.

```
package hu.domparse.D21YGM;

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

public class DomReadD21YGM {
   public static void main(String[] args) {
      try {
```

```
        // XML fájl betöltése
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setNamespaceAware(true);
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document document = builder.parse("XMLD21YGM.xml");

        // Normalizálás
        document.getDocumentElement().normalize();

        // Gyökér elem kiválasztása
        Element root = document.getDocumentElement();
        System.out.println("Gyökér elem: " + root.getNodeName());

        // Konzolra írás
        printNode(root, 0);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

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

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        System.out.print("<" + node.getNodeName());

        if (node.hasAttributes()) {
            for (int i = 0; i < node.getAttributes().getLength(); i++) {
                Node attribute = node.getAttributes().item(i);
                System.out.print(" " + attribute.getNodeName() + "=\"" + attribute.getNodeValue() + "\"");
            }
        }
        System.out.println(">");
    } else if (node.getNodeType() == Node.TEXT_NODE) {
        String text = node.getNodeValue().trim();
        if (!text.isEmpty()) {
            System.out.println(text);
        }
    }

    NodeList children = node.getChildNodes();
    for (int i = 0; i < children.getLength(); i++) {
        printNode(children.item(i), indent + 1);
    }

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        for (int i = 0; i < indent; i++) {
```

```
            System.out.print("  ");
        }
        System.out.println("</" + node.getNodeName() + ">");
    }
  }
}
```

3.2. Adatlekérdezés

A feladat teljesítésének érdekében négy darab lekérdezés készült el.

- 1. Írja ki az összes repülő gyártóját és típusát
- 2. Írja ki a repterek kódját, országát és városát.
- 3. Írja ki azon járatok adatait amellyek Airbus gyártmányú repülőt használnak
- 4. Írja ki azon kifutópályákat amellyek 3400 méternél hosszabbak.

```
package hu.domparse.D21YGM;

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

public class DomQueryD21YGM {
  public static void main(String[] args) {
    try {
      // XML fájl betöltése
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      factory.setNamespaceAware(true);
      DocumentBuilder builder = factory.newDocumentBuilder();
      Document document = builder.parse("XMLD21YGM.xml");

      // Normalizálás
      document.getDocumentElement().normalize();

      // Repülőgépek gyártója és típusa
      System.out.println("1. Lekérdezés");
      NodeList airplanes = document.getElementsByTagName("Repülő");
      for (int i = 0; i < airplanes.getLength(); i++) {
        Node airplane = airplanes.item(i);
        if (airplane.getNodeType() == Node.ELEMENT_NODE) {
          Element airplaneElement = (Element) airplane;
          String make = airplaneElement.getElementsByTagName("Gyártó").item(0).getTextContent();
          String model = airplaneElement.getElementsByTagName("Típus").item(0).getTextContent();
          System.out.println("Gyártó: " + make + ", Típus: " + model);
        }
```

```
        }
        System.out.println();

        // Repterek helye és IATA kódja
        System.out.println("2. Lekérdezés");
        NodeList airports = document.getElementsByTagName("Reptér");
        for (int i = 0; i < airports.getLength(); i++) {
            Node airport = airports.item(i);
            if (airport.getNodeType() == Node.ELEMENT_NODE) {
                Element airportElement = (Element) airport;
                String iata = airportElement.getAttribute("IATA");
                String country = airportElement.getElementsByTagName("Ország").item(0).getTextContent();
                String city = airportElement.getElementsByTagName("Város").item(0).getTextContent();
                System.out.println("IATA: " + iata + ", Ország: " + country + ", Város: " + city);
            }
        }
        System.out.println();

        // Járatok amellyek Airbus gyártmányú repülővel történnek
        System.out.println("3. Lekérdezés");
        for (int i = 0; i < airplanes.getLength(); i++) {
            Node airplane = airplanes.item(i);
            if (airplane.getNodeType() == Node.ELEMENT_NODE) {
                Element airplaneElement = (Element) airplane;
                String make = airplaneElement.getElementsByTagName("Gyártó").item(0).getTextContent();
                if ("Airbus".equalsIgnoreCase(make)) {
                    String callSign = airplaneElement.getAttribute("Hívójel");
                    NodeList routes = document.getElementsByTagName("Tartalmaz");
                    for (int j = 0; j < routes.getLength(); j++) {
                        Node route = routes.item(j);
                        if (route.getNodeType() == Node.ELEMENT_NODE) {
                            Element routeElement = (Element) route;
                            if (routeElement.getAttribute("Járat").equals(callSign)) {
                                String start = routeElement.getAttribute("Start");
                                String destination = routeElement.getAttribute("Cél");
                                System.out.println("Hívójel: " + callSign + ", Gyártó: " + make + ", Start: " + start + ", Cél:
" + destination);
                            }
                        }
                    }
                }
            }
        }
        System.out.println();


        // 3400 méternél hosszabb kifutópályák
        System.out.println("4. Lekérdezés");
        NodeList runways = document.getElementsByTagName("Kifutópálya");
```

```
        for (int i = 0; i < runways.getLength(); i++) {
            Node runway = runways.item(i);
            if (runway.getNodeType() == Node.ELEMENT_NODE) {
                Element runwayElement = (Element) runway;
                int length =
Integer.parseInt(runwayElement.getElementsByTagName("Hossz").item(0).getTextContent());
                if (length > 3400) {
                    String runwayId = runwayElement.getAttribute("id");
                    String airportIATA = runwayElement.getAttribute("fid");
                    for (int j = 0; j < airports.getLength(); j++) {
                        Node airport = airports.item(j);
                        if (airport.getNodeType() == Node.ELEMENT_NODE) {
                            Element airportElement = (Element) airport;
                            if (airportElement.getAttribute("IATA").equals(airportIATA)) {
                                String city =
airportElement.getElementsByTagName("Város").item(0).getTextContent();
                                System.out.println("Kifutópálya: " + runwayId + ", IATA: " + airportIATA + ", Város: " +
city + ", Hossz: " + length);
                            }
                        }
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

3.3. Adatmódosítás

Négy módosítást valósítottam meg:

```
package hu.domparse.D21YGM;

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

public class DomModifyD21YGM {
    public static void main(String[] args) {
        try {
            // XML fájl betöltése
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            factory.setNamespaceAware(true);
```

```
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document document = builder.parse("XMLD21YGM.xml");

        // Normalizálás
        document.getDocumentElement().normalize();

        // Módosítások végrehajtása
        modifyAirplaneModel(document);
        modifyCaptainNames(document);
        modifyFlightRoutes(document);
        modifyFlightDates(document);

        // Gyökér elem kiválasztása
        Element root = document.getDocumentElement();
        System.out.println("Gyökér elem: " + root.getNodeName());

        // Konzolra írás
        printNode(root, 0);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

// A400M Atlas típus cserélése A380-ra
private static void modifyAirplaneModel(Document document) {
    NodeList airplanes = document.getElementsByTagName("Repülő");
    for (int i = 0; i < airplanes.getLength(); i++) {
        Node airplane = airplanes.item(i);
        if (airplane.getNodeType() == Node.ELEMENT_NODE) {
            Element airplaneElement = (Element) airplane;
            Node modelNode = airplaneElement.getElementsByTagName("Típus").item(0);
            if (modelNode != null && "A400M Atlas".equals(modelNode.getTextContent())) {
                modelNode.setTextContent("A380");
            }
        }
    }
}

// Ha több mint 10000 órája van egy pilótának akkor kapjon doktori fokozatot
private static void modifyCaptainNames(Document document) {
    NodeList captains = document.getElementsByTagName("Kapitány");
    for (int i = 0; i < captains.getLength(); i++) {
        Node captain = captains.item(i);
        if (captain.getNodeType() == Node.ELEMENT_NODE) {
            Element captainElement = (Element) captain;
            int flightHours =
Integer.parseInt(captainElement.getElementsByTagName("Repült").item(0).getTextContent());
            if (flightHours > 10000) {
                Node firstNameNode = captainElement.getElementsByTagName("Vezetéknév").item(0);
```

```
            if (firstNameNode != null) {
                firstNameNode.setTextContent("Dr. " + firstNameNode.getTextContent());
            }
        }
    }
}

// Az összes járat átirányítása Bécsbe
private static void modifyFlightRoutes(Document document) {
    NodeList routes = document.getElementsByTagName("Tartalmaz");
    for (int i = 0; i < routes.getLength(); i++) {
        Node route = routes.item(i);
        if (route.getNodeType() == Node.ELEMENT_NODE) {
            Element routeElement = (Element) route;
            routeElement.setAttribute("Cél", "VIE");
        }
    }
}

// Jövőbe utazás
private static void modifyFlightDates(Document document) {
    NodeList flights = document.getElementsByTagName("Járat");
    for (int i = 0; i < flights.getLength(); i++) {
        Node flight = flights.item(i);
        if (flight.getNodeType() == Node.ELEMENT_NODE) {
            Element flightElement = (Element) flight;
            Node departureNode = flightElement.getElementsByTagName("Indulás").item(0);
            Node arrivalNode = flightElement.getElementsByTagName("Érkezés").item(0);

            if (departureNode != null) {
                departureNode.setTextContent(departureNode.getTextContent().replaceFirst("\\d{4}", "2025"));
            }
            if (arrivalNode != null) {
                arrivalNode.setTextContent(arrivalNode.getTextContent().replaceFirst("\\d{4}", "2025"));
            }
        }
    }
}


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

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        System.out.print("<" + node.getNodeName());
```

```
            if (node.hasAttributes()) {
                for (int i = 0; i < node.getAttributes().getLength(); i++) {
                    Node attribute = node.getAttributes().item(i);
                    System.out.print(" " + attribute.getNodeName() + "=\"" + attribute.getNodeValue() + "\"");
                }
            }
            System.out.println(">");
        } else if (node.getNodeType() == Node.TEXT_NODE) {
            String text = node.getNodeValue().trim();
            if (!text.isEmpty()) {
                System.out.println(text);
            }
        }

        NodeList children = node.getChildNodes();
        for (int i = 0; i < children.getLength(); i++) {
            printNode(children.item(i), indent + 1);
        }

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            for (int i = 0; i < indent; i++) {
                System.out.print("  ");
            }
            System.out.println("</" + node.getNodeName() + ">");
        }
    }
}
```

3.4.

```
package hu.domparse.D21YGM;

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;

import java.io.File;

public class DOMWriteD21YGM {
```

```java
public static void main(String[] args) {
    try {
        // XML fájl betöltése
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setNamespaceAware(true);
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document document = builder.parse("XMLD21YGM.xml");

        // Normalizálás
        document.getDocumentElement().normalize();

        // Gyökér elem kiválasztása
        Element root = document.getDocumentElement();
        System.out.println("Gyökér elem: " + root.getNodeName());

        // Konzolra írás
        printNode(root, 0);
        //Új fájlba mentés
        writeDocumentToFile(document, "XMLD21YGM1.xml");
    } catch (Exception e) {
        e.printStackTrace();
    }
}

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

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        System.out.print("<" + node.getNodeName());

        if (node.hasAttributes()) {
            for (int i = 0; i < node.getAttributes().getLength(); i++) {
                Node attribute = node.getAttributes().item(i);
                System.out.print(" " + attribute.getNodeName() + "=\"" + attribute.getNodeValue() + "\"");
            }
        }
        System.out.println(">");
    } else if (node.getNodeType() == Node.TEXT_NODE) {
        String text = node.getNodeValue().trim();
        if (!text.isEmpty()) {
            System.out.println(text);
        }
    }

    NodeList children = node.getChildNodes();
    for (int i = 0; i < children.getLength(); i++) {
        printNode(children.item(i), indent + 1);
```

```
        }

        if (node.getNodeType() == Node.ELEMENT_NODE) {
            for (int i = 0; i < indent; i++) {
                System.out.print("  ");
            }
            System.out.println("</" + node.getNodeName() + ">");
        }
    }
    private static void writeDocumentToFile(Document document, String outputFilePath) {
        try {
            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            DOMSource source = new DOMSource(document);
            StreamResult result = new StreamResult(new File(outputFilePath));
            transformer.transform(source, result);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```