

JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Autópálya felügyelet

Készítette: Babik Szilárd Kristóf

Neptunkód: A6NQW1

Dátum: 2022.11.24.

Tartalomjegyzék

A feladat leírása:.....	3
1. feladat	4
1a) Az adatbázis ER modellje:	4
1b) Az adatbázis konvertálása XDM modellre:	4
1c) Az XDM modell alapján XML dokumentum készítése:	4
1d) Az XML dokumentum alapján XMLSchema készítése (saját típusok, ref, key, keyref, speciális elemek):	9
2. feladat	13
2a) Adatolvasás:	13
2b) Adatlekérdezés:	14
2c) Adatmódosítás:	16

A feladat leírása:

A féléves feladatomban egy autópálya felügyelet (másnéven autópálya mérnökség) adatbázisát készítém el. A beadandóm elsősorban az autópályákon elkövetett gyorsajtásokkal foglalkozik, de tárolja a hatóságnál dolgozó alkalmazottakat és adataikat is, illetve magának a felügyeletnek a székhelyeinek az elhelyezkedéséről is tartalmaz információkat.

A feladatom ötletét az Adatbázisrendszerek I. című tárgy féléves feladata adta, kisebb-nagyobb módosításokkal.

E-mailes megegyezés alapján, a feladat 5 helyett 6 egyedet tartalmaz, viszont nem mindenhol van 4 tulajdonság.

Az egyedek és a köztük lévő relációk az alábbiak:

Az *AutopalyaFelugyelet* és a *Buntetes* egyedek közötti reláció 1:N, kötelező típusú kapcsolat, mivel a felügyelet több büntetést is kiszabhat.

A *Buntetes* és a *Szabalyserto* egyedek közötti reláció 1:1, kötelező típusú kapcsolat, ugyanis minden büntetésnek csak egy szabálysértője lehet, és egy szabálysértőnek egy alkalommal csak egy büntetést lehetséges kiszabni.

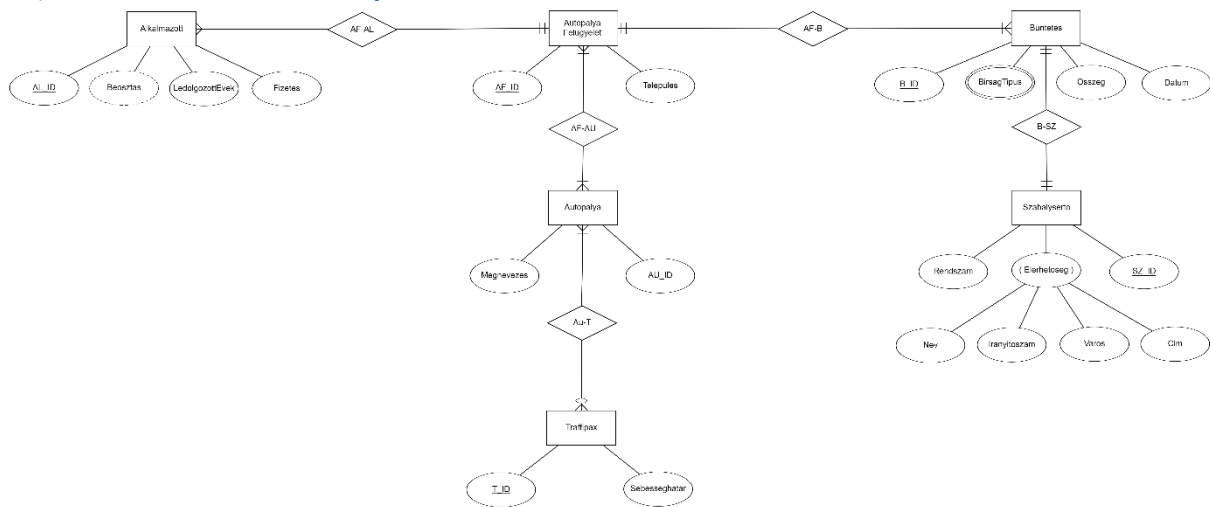
Az *AutopalyaFelugyelet* és az *Alkalmazott* egyedek közötti reláció 1:N, kötelező típusú kapcsolat, hiszen egy felügyelet létezik, aminek több alkalmazottja is lehet.

Az *AutopalyaFelugyelet* és az *Autopalya* egyedek közötti reláció szintén N:M, kötelező típusú kapcsolat, mivel egy felügyeletnek több autópálya is a hatáskörébe tartozhat, illetve ez fordítva is igaz, egy autópálya is tartozhat több felügyelethez is tartozhat.

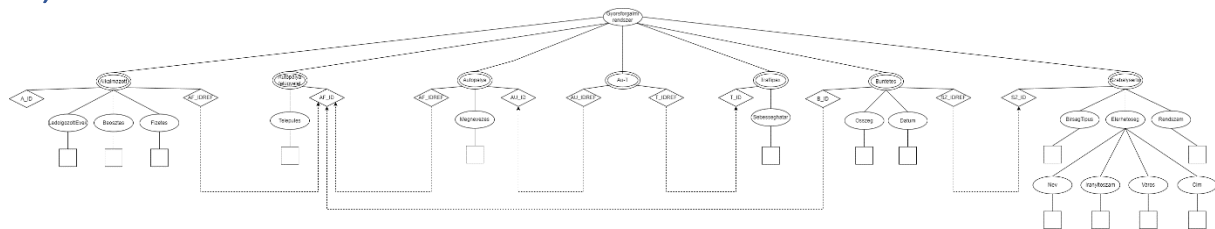
Az *Autopalya* és a *Traffipax* egyedek közötti reláció szintén 1:N típusú, de nem kötelező kapcsolat, mivel egy autópályán több sebességmérő kamera is elhelyezkedhet. Előfordulhat azonban olyan eset is, hogy az autópályán nincsen traffipax, ezért az *Autopalya* egyed részéről ez egy nem kötelező kapcsolat.

1. feladat

1a) Az adatbázis ER modellje:



1b) Az adatbázis konvertálása XDM modellre:



1c) Az XDM modell alapján XML dokumentum készítése:

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

<gyorsforgalmi_rendszer xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="XMLSchemaA6NQW1.xsd">

  <!-- Autópálya Felügyelet -->

  <autopalya_felugyelet AF_ID="1">
    <telepules>Budapest</telepules>
  </autopalya_felugyelet>

  <autopalya_felugyelet AF_ID="2">
    <telepules>Miskolc</telepules>
  </autopalya_felugyelet>

  <autopalya_felugyelet AF_ID="3">
    <telepules>Miskolc</telepules>
  </autopalya_felugyelet>

  <autopalya_felugyelet AF_ID="4">
    <telepules>Vásárosnamény</telepules>
```

```

</autopalya_felugyelet>

<autopalya_felugyelet AF_ID="5">
  <telepules>Törökbálint</telepules>
</autopalya_felugyelet>

<!-- Alkalmazottak -->

<alkalmazott AL_ID="1" AF_IDREF="1">
  <fizetes>650000</fizetes>
  <ledolgozott_evek>5</ledolgozott_evek>
</alkalmazott>

<alkalmazott AL_ID="2" AF_IDREF="4">
  <fizetes>180000</fizetes>
  <ledolgozott_evek>1</ledolgozott_evek>
</alkalmazott>

<alkalmazott AL_ID="3" AF_IDREF="2">
  <fizetes>1520000</fizetes>
  <ledolgozott_evek>12</ledolgozott_evek>
</alkalmazott>

<alkalmazott AL_ID="4" AF_IDREF="5">
  <fizetes>800000</fizetes>
  <ledolgozott_evek>8</ledolgozott_evek>
</alkalmazott>

<alkalmazott AL_ID="5" AF_IDREF="1">
  <fizetes>0</fizetes>
  <ledolgozott_evek>1</ledolgozott_evek>
</alkalmazott>

<alkalmazott AL_ID="6" AF_IDREF="3">
  <fizetes>420000</fizetes>
  <ledolgozott_evek>4</ledolgozott_evek>
</alkalmazott>

<!-- Autópályák -->

<autopalya AU_ID="1" AF_IDREF="1">
  <megnevezes>M0</megnevezes>
</autopalya>

<autopalya AU_ID="2" AF_IDREF="2">
  <megnevezes>M3</megnevezes>
</autopalya>

<autopalya AU_ID="3" AF_IDREF="5">

```

```

        <megnevezes>M5</megnevezes>
    </autopalya>

    <autopalya AU_ID="4" AF_IDREF="1">
        <megnevezes>M9</megnevezes>
    </autopalya>

    <autopalya AU_ID="5" AF_IDREF="4">
        <megnevezes>M34</megnevezes>
    </autopalya>

    <autopalya AU_ID="6" AF_IDREF="3">
        <megnevezes>M9</megnevezes>
    </autopalya>

    <!-- Sebességmérő kamerák -->

    <traffipax T_ID="1" AU_IDREF="3">
        <sebesseghatar>130</sebesseghatar>
    </traffipax>

    <traffipax T_ID="2" AU_IDREF="2">
        <sebesseghatar>90</sebesseghatar>
    </traffipax>

    <traffipax T_ID="3" AU_IDREF="5">
        <sebesseghatar>110</sebesseghatar>
    </traffipax>

    <traffipax T_ID="4" AU_IDREF="1">
        <sebesseghatar>130</sebesseghatar>
    </traffipax>

    <traffipax T_ID="5" AU_IDREF="6">
        <sebesseghatar>90</sebesseghatar>
    </traffipax>

    <traffipax T_ID="6" AU_IDREF="6">
        <sebesseghatar>130</sebesseghatar>
    </traffipax>

    <traffipax T_ID="7" AU_IDREF="4">
        <sebesseghatar>110</sebesseghatar>
    </traffipax>

    <!-- Kiszabott büntetések -->

    <buntetes B_ID="1" AF_IDREF="1">
        <osszeg>50000</osszeg>

```

```

        <datum>2022-03-02</datum>
    </buntetes>

    <buntetes B_ID="2" AF_IDREF="2">
        <osszeg>30000</osszeg>
        <datum>2021-12-10</datum>
    </buntetes>

    <buntetes B_ID="3" AF_IDREF="4">
        <osszeg>200000</osszeg>
        <datum>2020-11-20</datum>
    </buntetes>

    <buntetes B_ID="4" AF_IDREF="3">
        <osszeg>100000</osszeg>
        <datum>2019-06-15</datum>
    </buntetes>

    <buntetes B_ID="5" AF_IDREF="5">
        <osszeg>20000</osszeg>
        <datum>2021-01-01</datum>
    </buntetes>

    <!-- Szabálysértők és az adataik -->

    <szabalyserto SZ_ID="1" B_IDREF="1">
        <rendszam>TXT-637</rendszam>
        <elerhetoseg>
            <nev>Trab Antal</nev>
            <iranyitoszam>7542</iranyitoszam>
            <varos>Kisbajom</varos>
            <lakcim>Eötvös út 63</lakcim>
        </elerhetoseg>
    </szabalyserto>

    <szabalyserto SZ_ID="2" B_IDREF="2">
        <rendszam>GFD-874</rendszam>
        <elerhetoseg>
            <nev>Para Zita</nev>
            <iranyitoszam>8477</iranyitoszam>
            <varos>Veszprém</varos>
            <lakcim>Izabella utca 15 2/6</lakcim>
        </elerhetoseg>
    </szabalyserto>

    <szabalyserto SZ_ID="3" B_IDREF="3">
        <rendszam>HGF-321</rendszam>
        <elerhetoseg>
            <nev>Patta Nóra</nev>

```

```
        <iranyitoszam>2454</iranyitoszam>
        <varos>Iváncsa</varos>
        <lakcim>Erzsébet krt 43</lakcim>
    </elerhetoseg>
</szabalyserto>

<szabalyserto SZ_ID="4" B_IDREF="4">
    <rendszam>FSD-415</rendszam>
    <elerhetoseg>
        <nev>Fá Zoltán</nev>
        <iranyitoszam>4972</iranyitoszam>
        <varos>Gacsály</varos>
        <lakcim>Apor Péter utca 10</lakcim>
    </elerhetoseg>
</szabalyserto>

<szabalyserto SZ_ID="5" B_IDREF="5">
    <rendszam>UZR-543</rendszam>
    <elerhetoseg>
        <nev>Remek Elek</nev>
        <iranyitoszam>8692</iranyitoszam>
        <varos>Szőlősgyörök</varos>
        <lakcim>Bécsi utca 93</lakcim>
    </elerhetoseg>
</szabalyserto>

</gyorsforgalmi_rendszer>
```


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

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

  <!-- Elemek, tulajdonságok -->

  <xs:element name="telepules" type="xs:string" />
  <xs:element name="fizetes" type="xs:integer" />
  <xs:element name="ledolgozott_evek" type="xs:integer" />
  <xs:element name="megnevezes" type="xs:string" />
  <xs:element name="sebesseghatar" type="xs:integer" />
  <xs:element name="osszeg" type="xs:integer" />
  <xs:element name="datum" type="xs:date" />
  <xs:element name="rendszam" type="xs:string" />
  <xs:element name="nev" type="xs:string" />
  <xs:element name="iranyitoszam" type="xs:integer" />
  <xs:element name="varos" type="xs:string" />
  <xs:element name="lakcim" type="xs:string" />

  <xs:attribute name="AF_ID" type="xs:integer" />
  <xs:attribute name="AF_IDREF" type="xs:integer" />
  <xs:attribute name="AL_ID" type="xs:integer" />
  <xs:attribute name="AL_IDREF" type="xs:integer" />
  <xs:attribute name="AU_ID" type="xs:integer" />
  <xs:attribute name="AU_IDREF" type="xs:integer" />
  <xs:attribute name="T_ID" type="xs:integer" />
  <xs:attribute name="B_ID" type="xs:integer" />
  <xs:attribute name="B_IDREF" type="xs:integer" />
  <xs:attribute name="SZ_ID" type="xs:integer" />

  <!-- Komplex típusok -->

  <xs:complexType name="autopalya_felugyelet_tipus">
    <xs:sequence>
      <xs:element ref="telepules"/>
    </xs:sequence>
    <xs:attribute ref="AF_ID" use="required"/>
  </xs:complexType>

  <xs:complexType name="alkalmazott_tipus">
    <xs:sequence>
      <xs:element ref="fizetes"/>
      <xs:element ref="ledolgozott_evek"/>
    </xs:sequence>
    <xs:attribute ref="AL_ID" use="required"/>
    <xs:attribute ref="AF_IDREF" use="required"/>
  </xs:complexType>
</xs:schema>
```

```

</xs:complexType>

<xs:complexType name="autopalya_tipus">
  <xs:sequence>
    <xs:element ref="megnevezes"/>
  </xs:sequence>
  <xs:attribute ref="AU_ID" use="required"/>
  <xs:attribute ref="AF_IDREF" use="required"/>
</xs:complexType>

<xs:complexType name="traffipax_tipus">
  <xs:sequence>
    <xs:element ref="sebesseghatar"/>
  </xs:sequence>
  <xs:attribute ref="T_ID" use="required"/>
  <xs:attribute ref="AU_IDREF" use="required"/>
</xs:complexType>

<xs:complexType name="buntetes_tipus">
  <xs:sequence>
    <xs:element ref="osszeg"/>
    <xs:element ref="datum"/>
  </xs:sequence>
  <xs:attribute ref="B_ID" use="required"/>
  <xs:attribute ref="AF_IDREF" use="required"/>
</xs:complexType>

<xs:complexType name="elerhetoseg_tipus">
  <xs:sequence>
    <xs:element ref="nev"/>
    <xs:element ref="iranyitoszam"/>
    <xs:element ref="varos"/>
    <xs:element ref="lakcim"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="szabalyserto_tipus">
  <xs:sequence>
    <xs:element ref="rendszer"/>
    <xs:element name="elerhetoseg" type="elerhetoseg_tipus"/>
  </xs:sequence>
  <xs:attribute ref="SZ_ID" use="required"/>
  <xs:attribute ref="B_IDREF" use="required"/>
</xs:complexType>

<!-- Kiszabott büntetések -->

<xs:element name="gyorsforgalmi_rendszer">
  <xs:complexType>

```

```

        <xs:sequence>
            <xs:element name="autopalya_felugyelet"
type="autopalya_felugyelet_tipus" maxOccurs="unbounded"/>
            <xs:element name="alkalmazott" type="alkalmazott_tipus"
maxOccurs="unbounded"/>
            <xs:element name="autopalya" type="autopalya_tipus"
maxOccurs="unbounded"/>
            <xs:element name="traffipax" type="traffipax_tipus"
maxOccurs="unbounded"/>
            <xs:element name="buntetes" type="buntetes_tipus"
maxOccurs="unbounded"/>
            <xs:element name="szabalyserto" type="szabalyserto_tipus"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <!-- Kulcsok -->

    <xs:key name="AF_ID">
        <xs:selector xpath="autopalya_felugyelet"/>
        <xs:field xpath="@AF_ID"/>
    </xs:key>

    <xs:key name="AL_ID">
        <xs:selector xpath="alkalmazott"/>
        <xs:field xpath="@AL_ID"/>
    </xs:key>

    <xs:key name="AU_ID">
        <xs:selector xpath="autopalya"/>
        <xs:field xpath="@AU_ID"/>
    </xs:key>

    <xs:key name="T_ID">
        <xs:selector xpath="traffipax"/>
        <xs:field xpath="@T_ID"/>
    </xs:key>

    <xs:key name="B_ID">
        <xs:selector xpath="buntetes"/>
        <xs:field xpath="@B_ID"/>
    </xs:key>

    <xs:key name="SZ_ID">
        <xs:selector xpath="szabalyserto"/>
        <xs:field xpath="@SZ_ID"/>
    </xs:key>

    <!-- Kulcshivatkozások (idegen kulcsok) -->

```

```

<xs:keyref name="autopalya_felugyelet_FK1" refer="AF_ID">
  <xs:selector xpath="alkalmazott"></xs:selector>
  <xs:field xpath="@AF_IDREF"></xs:field>
</xs:keyref>

<xs:keyref name="autopalya_felugyelet_FK2" refer="AF_ID">
  <xs:selector xpath="autopalya"></xs:selector>
  <xs:field xpath="@AF_IDREF"></xs:field>
</xs:keyref>

<xs:keyref name="autopalya_felugyelet_FK3" refer="AF_ID">
  <xs:selector xpath="buntetes"></xs:selector>
  <xs:field xpath="@AF_IDREF"></xs:field>
</xs:keyref>

<xs:keyref name="autopalya_FK1" refer="AU_ID">
  <xs:selector xpath="traffipax"></xs:selector>
  <xs:field xpath="@AU_IDREF"></xs:field>
</xs:keyref>

<xs:keyref name="buntetes_FK1" refer="B_ID">
  <xs:selector xpath="szabalyserto"></xs:selector>
  <xs:field xpath="@B_IDREF"></xs:field>
</xs:keyref>

</xs:element>

</xs:schema>

```

2. feladat

2a) Adatolvasás:

```
package hu.domparsing.a6nqw1;

import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.w3c.dom.Text;
import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import java.io.File;
import java.io.IOException;

public class DomReadA6Nqw1 {

    public static void main(String[] args) {

        //Importing and parsing XML file
        File xmlFile = new File("XMLA6Nqw1.xml");
        Document doc = introduceFile(xmlFile);

        if (doc != null) {
            doc.getDocumentElement().normalize();
            System.out.println("Root element: " +
doc.getDocumentElement().getNodeName());
        } else {
            System.out.println("Document is null");
            System.exit(-1);
        }

        //Reading all nodes of the XML file
        NodeList nodeList = doc.getDocumentElement().getChildNodes();
        String indent = "";
        listData(nodeList, indent);
    }

    public static Document introduceFile (File xmlFile){
        Document doc = null;

        try{
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dbBuilder = dbFactory.newDocumentBuilder();
            doc = dbBuilder.parse(xmlFile);
        } catch (ParserConfigurationException | SAXException | IOException
e) {
            e.printStackTrace();
        }
        return doc;
    }

    public static void listData(NodeList nodeList, String indent){
        indent += "\t";

        if(nodeList != null) {
```

```

        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE &&
!node.getTextContent().trim().isEmpty()) {
                System.out.println(indent + "{" + node.getNodeName() +
":}");

                NodeList nodeList_new = node.getChildNodes();
                listData(nodeList_new, indent);
            } else if (node instanceof Text) {
                String value = node.getNodeValue().trim();
                if (value.isEmpty()) {
                    continue;
                }
                System.out.println(indent + node.getTextContent());
            }
        }
    }
}
}
}

```

2b) Adatlekérdezés:

```

package hu.domparsing.a6nqw1;

import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.w3c.dom.Text;
import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import java.io.File;
import java.io.IOException;

public class DomQueryA6NQW1 {
    public static void main(String[] args) {

        //Importing and parsing XML file
        File xmlFile = new File("XMLA6NQW1.xml");
        Document doc = introduceFile(xmlFile);

        if (doc != null) {
            doc.getDocumentElement().normalize();
        } else {
            System.out.println("Document is null");
            System.exit(-1);
        }

        //Processing the query on the XML file
        NodeList queryList =
doc.getDocumentElement().getElementsByTagName("buntetes");

        //30000 Ft feletti büntetési tételek kigyűjtése
        for (int i = 0; i < queryList.getLength(); i++) {
            NodeList query = queryList.item(i).getChildNodes();
            for (int j = 0; j < query.getLength(); j++) {
                if (query.item(j).getNodeName().equals("osszeg") &&
Integer.parseInt(query.item(j).getTextContent()) > 30000) {
                    System.out.println("{buntetes}");
                }
            }
        }
    }
}

```

```

        listData(queryList.item(i).getChildNodes(), "");
    }
}

}

}

public static Document introduceFile (File xmlFile){
    Document doc = null;

    try {
        DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
        DocumentBuilder dbBuilder = dbFactory.newDocumentBuilder();
        doc = dbBuilder.parse(xmlFile);
    } catch (ParserConfigurationException | SAXException | IOException
e) {
        e.printStackTrace();
    }
    return doc;
}

public static void listData(NodeList nodeList, String indent){
    indent += "\t";

    if (nodeList != null) {
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE &&
!node.getTextContent().trim().isEmpty()) {
                System.out.println(indent + "{" + node.getNodeName() +
":");

                NodeList nodeList_new = node.getChildNodes();
                listData(nodeList_new, indent);
            } else if (node instanceof Text){
                String value = node.getNodeValue().trim();
                if (value.isEmpty()){
                    continue;
                }
                System.out.println(indent + node.getTextContent());
            }
        }
    }
}
}
}
}

```

2c) Adatmódosítás:

```
package hu.domparsing.a6nqw1;

import java.io.File;
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.*;
import org.xml.sax.SAXException;

public class DomModifyA6NQW1 {
    public static void main(String[] args) {

        //Importing and parsing XML file
        File xmlFile = new File("XMLA6NQW1.xml");
        Document doc = introduceFile(xmlFile);

        if (doc != null) {
            doc.getDocumentElement().normalize();
            System.out.println("Root element: " +
doc.getDocumentElement().getNodeName());
        } else {
            System.out.println("Document is null");
            System.exit(-1);
        }

        //Changing "osszeg" values by deducting 2500 from the penalty value
        NodeList modifyList =
doc.getDocumentElement().getElementsByTagName("osszeg");
        modifyData(modifyList);

        //Querying for "buntetes", where the changed values are visible
        NodeList queryList =
doc.getDocumentElement().getElementsByTagName("buntetes");
        for (int i = 0; i < queryList.getLength(); i++) {
            NodeList query = queryList.item(i).getChildNodes();
            for (int j = 0; j < query.getLength(); j++) {
                if (query.item(j).getNodeName().equals("osszeg")){
                    System.out.println("{buntetes}");
                    listData(queryList.item(i).getChildNodes(), "");
                }
            }
        }
    }

    public static Document introduceFile (File xmlFile){
        Document doc = null;

        try{
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dbBuilder = dbFactory.newDocumentBuilder();
            doc = dbBuilder.parse(xmlFile);
        } catch (ParserConfigurationException | SAXException | IOException
e) {
            e.printStackTrace();
        }
        return doc;
    }
}
```



```

    }

    public static void listData(NodeList nodeList, String indent){
        indent += "\t";

        if(nodeList != null) {
            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);
                if (node.getNodeType() == Node.ELEMENT_NODE &&
!node.getTextContent().trim().isEmpty()) {
                    System.out.println(indent + "{" + node.getNodeName() +
":");

                    NodeList nodeList_new = node.getChildNodes();
                    listData(nodeList_new, indent);
                } else if (node instanceof Text){
                    String value = node.getNodeValue().trim();
                    if (value.isEmpty()){
                        continue;
                    }
                    System.out.println(indent + node.getTextContent());
                }
            }
        }
    }

    public static void modifyData(NodeList nodeList){
        if (nodeList != null){
            for (int i = 0; i < nodeList.getLength(); i++) {
                int penalty_value =
Integer.parseInt(nodeList.item(i).getTextContent());
                penalty_value -= 2500;

nodeList.item(i).setTextContent(Integer.toString(penalty_value));
            }
        }
    }
}

```