Jegyzőkönyv

Adatkezelés XML környezetben

Féléves feladat

Készítette: **Polonkai Dávid**
Neptunkód: **GPNWZT**

Tartalomjegyzék

1. Feladat leírása

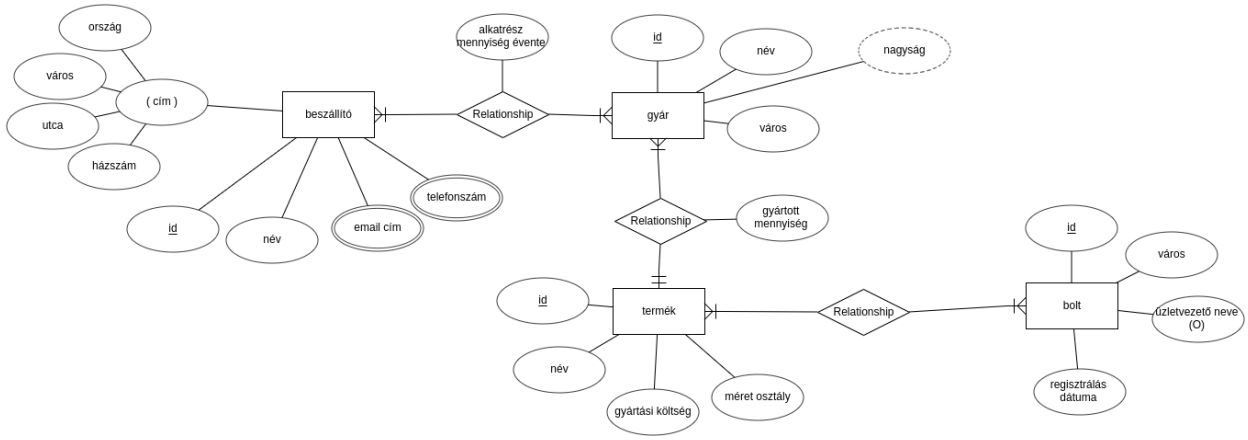
A beadandó feladatomat egy kisebb rendszerre alapozom. A rendszer alapja egy egyszerű gyár, anyagbeszállító, termék, bolt, illetve a közöttük levő kapcsolatokat dolgozza fel. Az adatok tárolása az XML jelölőnyelvben történik. A beadandóm első részében erről az adatszerkezetről lesz szó. A második részben az adatokat beolvasni képes java nyelvű `DOM` program. Az adatrendszer szerkezete: *Az XML-ben a tulajdonság adattagokat* □*-vel jelöltem*

1. beszállítók:
   - □ id: A beszállító cég azonosítására szolgál
   - nev: A beszállító cég neve
   - email: A beszállító email címe
   - telefonszam: A beszállító telefonszáma
   - cim: A beszállító címe (ország, város, utca, házszám)
2. gyárak:
   - □ id: A gyár azonosítására szolgál
   - □ t_id: A gyár által gyártott terméket azonosítja
   - nev: A gyár neve
   - nagysag: Egy szervezeten belüli nagyságérték, amely megmondja az adott gyár nagyságát
   - varos: A gyár helyének azonosítására szolgál
   - gyartott_mennyiseg: Az összesen gyártott termék mennyiségét adja meg
3. termékek:
   - □ id: Az adott termékre utal
   - nev: A termék neve
   - gyartasikoltseg: A termék gyártási költsége
   - meretosztaly: A termék méretére utaló adattag
4. boltok:
   - □ id: A bolt azonosítója
   - varos: A bolt székhelye
   - uzletvezeto: Az üzletvezető neve
   - regisztralasdatuma: A bolttal való kereskedés kezdete
5. k_besz_gyar: A beszállító és gyár közötti kapcsolatot tárolja
   - □ id: A kapcsolatot azonosító attribútum
   - □ idbesz: Elem, melynek egyetlen attribútuma a refID ez a beszállítóra utal
   - □ idgyar: Elem, melynek egyetlen attribútuma a refID ez a gyárra utal
   - alkatreszmennyiseg: A beszállítótól gyárba érkező alkatrészmennyiségre utal
6. k_term_bolt:
   - □ id: A kapcsolatot azonosító attribútum
   - □ idterm: Elem, melynek egyetlen attribútuma a refID ez a termékre utal
   - □ idbolt: Elem, melynek egyetlen attribútuma a refID ez a boltra utal

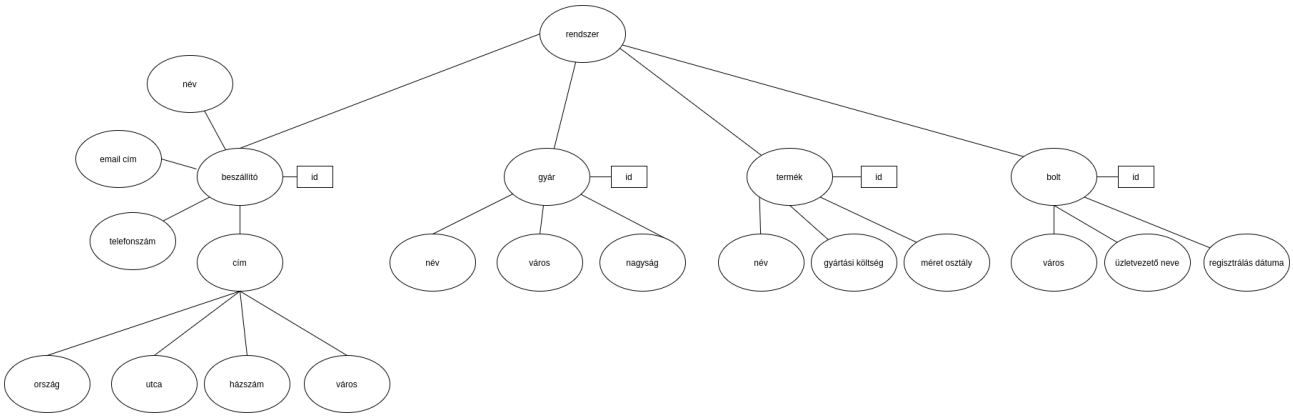
2. A feladat első része

Ebben a részben az ER modellt, XDM modellt, XML és XMLSchema dokumentumokat ismertetem.

2.1. ER modell



2.2. XDM modell



2.3. XML kód

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><?xml-model
    href="XMLSchemagpnwzt.xsd" type="application/xml" schematypens="
    http://www.w3.org/2001/XMLSchema"?><rendszer>
  <beszallitok>
    <beszallito id="be1">
      <nev>fa.zrt</nev>
      <email>fa@fa.hu</email>
      <telefonszam>0636778988</telefonszam>
      <cim>
        <orszag>Magyarország</orszag>
        <varos>Miskolc</varos>
        <utca>Jo utca 1</utca>
        <hazszam>12</hazszam>
```

```
      </cim>
    </beszallito>
  </beszallitok>
  <gyarak>
    <gyar id="gy1" t_id="t1">
      <nev>A1</nev>
      <nagysag>200</nagysag>
      <varos>Miskolc</varos>
      <gyartott_mennyiseg>300</gyartott_mennyiseg>
    </gyar>
    <gyar id="gy2" t_id="t2">
      <nev>A2</nev>
      <nagysag>220</nagysag>
      <varos>Ózd</varos>
      <gyartott_mennyiseg>120</gyartott_mennyiseg>
    </gyar>
  </gyarak>
  <termekek>
    <termek id="t1">
      <nev>asztal</nev>
      <gyartasikoltseg>120</gyartasikoltseg>
      <meretosztaly>5</meretosztaly>
    </termek>
    <termek id="t2">
      <nev>szek</nev>
      <gyartasikoltseg>50</gyartasikoltseg>
      <meretosztaly>2</meretosztaly>
    </termek>
  </termekek>
  <boltok>
    <bolt id="bo1">
      <varos>Miskolc</varos>
      <uzletvezeto>Nagy Miklos</uzletvezeto>
      <regisztalasdatuma>2018</regisztalasdatuma>
    </bolt>
  </boltok>
  <k_besz_gyar>
    <kapcsolat id="ka1">
      <idbesz refID="be1" />
      <idgyar refID="gy1" />
      <alkatreszmennyiseg>2000</alkatreszmennyiseg>
    </kapcsolat>
    <kapcsolat id="ka2">
      <idbesz refID="be1" />
      <idgyar refID="gy2" />
      <alkatreszmennyiseg>200</alkatreszmennyiseg>
    </kapcsolat>
  </k_besz_gyar>
  <k_term_bolt>
    <kapcsolat id="kb1">
```

```
      <idterm refID="t1"/>
      <idbolt refID="bo1"/>
    </kapcsolat>
  </k_term_bolt>
</rendszer>
```

2.4. XMLSchema kód

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
  <xs:element name="rendszer">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="beszallitok" type="beszallitoktype"
    maxOccurs="1"/>
        <xs:element name="gyarak" type="gyaraktype" maxOccurs="1"/>
        <xs:element name="termekek" type="termekektype" maxOccurs="1
    "/>
        <xs:element name="boltok" type="boltoktype" maxOccurs="1"/>
        <xs:element name="k_besz_gyar" type="k_besz_gyartype"
    maxOccurs="1"/>
        <xs:element name="k_term_bolt" type="k_term_bolttype"
    maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="beszallitoktype">
    <xs:sequence>
      <xs:element name="beszallito" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="nev" type="xs:string"/>
            <xs:element name="email" type="xs:string"/>
            <xs:element name="telefonszam" type="xs:string"/>
            <xs:element name="cim" type="cimtype"/>
          </xs:sequence>
          <xs:attribute name="id" type="xs:ID" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>



  <xs:complexType name="cimtype">
    <xs:sequence>
      <xs:element name="orszag" type="xs:string"/>
      <xs:element name="varos" type="xs:string"/>
      <xs:element name="utca" type="xs:string"/>
```

```
    <xs:element name="hazszam" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="gyaraktype">
  <xs:sequence>
    <xs:element name="gyar" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="nev" type="xs:string"/>
          <xs:element name="nagysag" type="xs:integer"/>
          <xs:element name="varos" type="xs:string"/>
          <xs:element name="gyartott_mennyiseg" type="xs:integer"/
 >
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required"/>
        <xs:attribute name="t_id" type="xs:IDREF"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="termekektype">
  <xs:sequence>
    <xs:element name="termek" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="nev" type="xs:string"/>
          <xs:element name="gyartasikoltseg" type="xs:integer"/>
          <xs:element name="meretosztaly" type="xs:integer"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="boltoktype">
  <xs:sequence>
    <xs:element name="bolt" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="varos" type="xs:string"/>
          <xs:element name="uzletvezeto" type="xs:string"
 minOccurs="0"/>
          <xs:element name="regisztalasdatuma" type="xs:integer"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="required"/>
      </xs:complexType>
    </xs:element>
```

```
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="k_besz_gyartype">
    <xs:sequence>
      <xs:element name="kapcsolat" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="idbesz">
              <xs:complexType>
                <xs:attribute name="refID" type="xs:IDREF"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="idgyar">
              <xs:complexType>
                <xs:attribute name="refID" type="xs:IDREF"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="alkatreszmennyiseg" type="xs:integer"/
   >
          </xs:sequence>
          <xs:attribute name="id" type="xs:ID" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="k_term_bolttype">
    <xs:sequence>
      <xs:element name="kapcsolat" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="idterm">
              <xs:complexType>
                <xs:attribute name="refID" type="xs:IDREF"/>
              </xs:complexType>
            </xs:element>
            <xs:element name="idbolt">
              <xs:complexType>
                <xs:attribute name="refID" type="xs:IDREF"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="id" type="xs:ID" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

3. A feladat második része

3.1. Adatolvasás

```
package hu.domparse.gpnwzt;

import java.io.IOException;
import java.util.ArrayList;

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

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

public class DOMReadgpnwzt {
  private Node root;
  private Document document;

  public static void main(String[] args) {
    try {
      DOMReadgpnwzt domreader = new DOMReadgpnwzt();
      domreader.gyarTermek();
      // domreader.beszallitogyarto();
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
  }

  // beállítja a document és a root elemeket
  public DOMReadgpnwzt() throws SAXException, IOException,
   ParserConfigurationException {
    DocumentBuilderFactory documentBuilderFactory =
   DocumentBuilderFactory.newInstance();
    DocumentBuilder documentBuilder = documentBuilderFactory.
   newDocumentBuilder();
    this.document = documentBuilder.parse("XMLgpnwzt.xml");
    this.document.getDocumentElement().normalize();
    this.root = document.getDocumentElement();
  }

  // kilistázza az xml-ben levő adatokat rendszerezve
  public void listAll() {
    NodeList nodeList = this.root.getChildNodes();
    for (int i = nodeList.getLength() - 1; i >= 0; i--) {
```

```
    if (nodeList.item(i).getNodeType() == Node.ELEMENT_NODE) {
      System.out.println();
      System.out.println();
      System.out.println(nodeList.item(i).getNodeName().toString()
.toUpperCase());
      NodeList subNodeList = nodeList.item(i).getChildNodes();
      for (int j = 0; j < subNodeList.getLength(); j++) {
        Node currentsubnode = subNodeList.item(j);
        if (currentsubnode.getNodeType() == Node.ELEMENT_NODE) {
          NamedNodeMap att = currentsubnode.getAttributes();
          String current = "";
          for (int k = 0; k < att.getLength(); k++) {
            Node attnode = att.item(k);
            if (attnode.getNodeType() == Node.ATTRIBUTE_NODE) {
              if (j == 1) {
                System.out.print(attnode.getNodeName() + " | ");
              }
              current += attnode.getTextContent() + " | ";
            }
          }
          NodeList subsubNodeList = currentsubnode.getChildNodes()
;
          for (int k = 0; k < subsubNodeList.getLength(); k++) {
            Node currentsubsubnode = subsubNodeList.item(k);
            if (currentsubsubnode.getNodeType() == Node.
ELEMENT_NODE) {
              if (j == 1 && currentsubsubnode.getNodeName() != "
cim") {
                System.out.print(currentsubsubnode.getNodeName() +
 " | ");
              }
              switch (currentsubsubnode.getNodeName()) {
                case "idterm":
                case "idbolt":
                case "idbesz":
                case "idgyar":
                  current += currentsubsubnode.getAttributes().
item(0).getTextContent() + " | ";
                  break;
                case "cim": {
                  NodeList cimNode = currentsubsubnode.
getChildNodes();
                  for (int l = 0; l < cimNode.getLength(); l++) {
                    if (cimNode.item(l).getNodeType() == Node.
ELEMENT_NODE) {
                      if (j == 1) {
                        System.out.print(cimNode.item(l).
getNodeName() + " | ");
                      }
                      current += cimNode.item(l).getTextContent()
```

```
 + "␣|␣";
                    }
                  }
                }
                  break;
                default:
                  current += currentsubsubnode.getTextContent() +
 "␣|␣";
              }
            }
          }
          System.out.println();
          System.out.print(current);
        }
      }
    }
  }
}

// A gyárat és a gyár által gyártot terméket írja ki
public void gyarTermek() throws DOMException, Exception {
  NodeList gyarak = this.document.getElementsByTagName("gyar");
  for (int i = 0; i < gyarak.getLength(); i++) {
    NodeList currentnodelist = gyarak.item(i).getChildNodes();
    Node ref = findIdInDoc(gyarak.item(i).getAttributes().
 getNamedItem("t_id").getTextContent());
    NodeList termek = ref.getChildNodes();
    String current = "";
    for (int j = 0; j < currentnodelist.getLength(); j++) {
      Node currentnode = currentnodelist.item(j);
      if (currentnode.getNodeType() == Node.ELEMENT_NODE) {
        if (i == 0)
          System.out.print(currentnodelist.item(1).getParentNode()
 .getNodeName() + ":"
              + currentnode.getNodeName() + "␣|␣");
        current += currentnode.getTextContent() + "␣|␣";
      }
    }
    for (int j = 0; j < termek.getLength(); j++) {
      Node currentnode = termek.item(j);
      if (currentnode.getNodeType() == Node.ELEMENT_NODE) {
        if (i == 0)
          System.out.print(
              termek.item(1).getParentNode().getNodeName() + ":" +
  currentnode.getNodeName() + "␣|␣");
        current += currentnode.getTextContent() + "␣|␣";
      }
    }
    System.out.println();
    System.out.print(current);
```

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

  // A gyártót és a hozzá tartozó beszállítót írja ki, a beszállító
   ciméből csak
  // az országot veszi
  public void beszallitogyarto() throws Exception {
    Node node = this.document.getElementsByTagName("k_besz_gyar").
   item(0);
    NodeList kapcsolatok = node.getChildNodes();
    System.out.print("alkatreszmennyiseg" + " | ");
    for (int i = 0; i < kapcsolatok.getLength(); i++) {
      if (kapcsolatok.item(i).getNodeType() == Node.ELEMENT_NODE) {
        NodeList kapcsolat = kapcsolatok.item(i).getChildNodes();
        String beszref = kapcsolat.item(1).getAttributes().
   getNamedItem("refID").getTextContent();
        String gyarref = kapcsolat.item(3).getAttributes().
   getNamedItem("refID").getTextContent();
        String alkmennyiseg = kapcsolat.item(5).getTextContent();
        NodeList nl1 = findIdInDoc(beszref).getChildNodes();
        NodeList nl2 = findIdInDoc(gyarref).getChildNodes();
        String current = "";
        String nl1name = nl1.item(1).getParentNode().getNodeName();
        String nl2name = nl2.item(1).getParentNode().getNodeName();
        for (int j = 0; j < nl1.getLength(); j++) {
          Node currentnode = nl1.item(j);
          if (currentnode.getNodeType() == Node.ELEMENT_NODE) {
            if (currentnode.getNodeName().equals("cim")) {
              if (i == 1)
                System.out.print(
                    nl1name + ":" + currentnode.getChildNodes().item
   (1).getNodeName() + " | ");
              current += currentnode.getChildNodes().item(1).
   getTextContent() + " | ";
            } else {
              if (i == 1)
                System.out.print(nl1name + ":" + currentnode.
   getNodeName() + " | ");
              current += currentnode.getTextContent() + " | ";
            }
          }
        }
        for (int j = 0; j < nl2.getLength(); j++) {
          Node currentnode = nl2.item(j);
          if (currentnode.getNodeType() == Node.ELEMENT_NODE) {
            if (i == 1)
              System.out.print(nl2name + ":" + currentnode.
   getNodeName() + " | ");
            current += currentnode.getTextContent() + " | ";
```

```
          }
        }
        System.out.println();
        System.out.print(alkmennyiseg + " | ");
        System.out.print(current);
      }
    }
    System.out.println();
  }

  // Ez a metódus megkeresi a megadott ID-jű adattagot, hibát dob,
   ha a
  // hivatkozott adattag nem létezik
  private Node findIdInDoc(String id) throws Exception {
    NodeList nl1 = this.root.getChildNodes();
    for (int i = 0; i < nl1.getLength(); i++) {
      if (nl1.item(i).getNodeType() == Node.ELEMENT_NODE) {
        for (int j = 0; j < nl1.item(i).getChildNodes().getLength();
    j++) {
          if (nl1.item(i).getChildNodes().item(j).getNodeType() ==
   Node.ELEMENT_NODE) {
            for (int k = 0; k < nl1.item(i).getChildNodes().item(j).
   getAttributes().getLength(); k++) {
              if (nl1.item(i).getChildNodes().item(j).getAttributes
   ().item(k).getTextContent().equals(id)
                  && nl1.item(i).getChildNodes().item(j).
   getAttributes().item(k).getNodeName()
                        .equals("id")) {
                return nl1.item(i).getChildNodes().item(j);
              }
            }
          }
        }
      }
    }
    throw new Exception("Az id vagy a referencia nem létezik");
  }

  //a módosításhoz kilistázza a módosítható nodeokat
  public void listForModify() {
    ArrayList<NodeList> arraynodelist = new ArrayList<NodeList>();
    arraynodelist.add(this.document.getElementsByTagName("gyar"));
    arraynodelist.add(this.document.getElementsByTagName("termek"));
    arraynodelist.add(this.document.getElementsByTagName("bolt"));
    for (NodeList nodelist : arraynodelist) {
      for (int i = 0; i < nodelist.getLength(); i++) {
        Node node = nodelist.item(i);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
          System.out.print("id: " + node.getAttributes().
   getNamedItem("id").getTextContent() + " | tipus: "
```

```
                + node.getNodeName() + " | ");
            NodeList nodeList2 = node.getChildNodes();

            for (int j = 0; j < nodeList2.getLength(); j++) {
              Node node2 = nodeList2.item(j);
              if (node2.getNodeType() == Node.ELEMENT_NODE) {
                System.out.print(node2.getNodeName()+":"+node2.
    getTextContent()+" | ");
              }
            }
            System.out.println();
          }
        }
      }

    }
}
```

3.2. Adatolvasás

```
package hu.domparse.gpnwzt;

import java.io.File;
import java.io.IOException;
import java.util.Scanner;

import javax.xml.XMLConstants;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
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 javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.validation.Validator;

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

public class DOMModifygpnwzt {
  private Node root;
  private Document document;
  private Scanner in;

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

```
      DOMModifygpnwzt domModifier = new DOMModifygpnwzt();
      domModifier.selector();
      domModifier.destructor();
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
}

// Beállítja a document és a root elemeket
public DOMModifygpnwzt() throws SAXException, IOException,
 ParserConfigurationException {
    DocumentBuilderFactory documentBuilderFactory =
 DocumentBuilderFactory.newInstance();
    DocumentBuilder documentBuilder = documentBuilderFactory.
 newDocumentBuilder();
    this.document = documentBuilder.parse("XMLgpnwzt.xml");
    this.document.getDocumentElement().normalize();
    this.root = document.getDocumentElement();
    this.in = new Scanner(System.in);
}

// A destructor az adatok kiírását és a schema alapján történő
 validációt végzi
public void destructor() throws TransformerException, SAXException
 , IOException {
    String language = XMLConstants.W3C_XML_SCHEMA_NS_URI;
    SchemaFactory factory = SchemaFactory.newInstance(language);
    Schema schema = factory.newSchema(new File("XMLSchemagpnwzt.xsd"
 ));
    Validator validator = schema.newValidator();
    DOMSource domsource = new DOMSource(this.document);
    validator.validate(domsource);

    TransformerFactory transformerFactory = TransformerFactory.
 newInstance();
    Transformer transformer = transformerFactory.newTransformer();
    StreamResult result = new StreamResult(new File("XMLgpnwzt.xml")
 );
    transformer.transform(domsource, result);
}

// ez a függvény meg a módosítandó nodeot, ellenőrzi hogy lehetsé
 ges-e a
// módosítás és elvégzi
public void modify(String id) throws Exception {
    Node node = findIdInDoc(id);
    String name = node.getNodeName();
    if (name.equals("beszallito") || name.equals("kapcsolat"))
      throw new Exception("Nem lehetséges módosítani a kért node-ot"
 );
```

```
    NodeList nl = node.getChildNodes();
    for (int i = 0; i < nl.getLength(); i++) {
      Node current = nl.item(i);
      if (current.getNodeType() == Node.ELEMENT_NODE) {
        current.setTextContent(readModification(current.getNodeName
 (), current.getTextContent()));
      }
    }
  }

  // beolvassa az új adatot
  public String readModification(String nodename, String old) {
    System.out.print("Adja meg az új értéket (régi érték: " + old +
 "): ");
    String ret = in.nextLine();
    return ret;
  }

  // Ez a metódus megkeresi a megadott ID-jű adattagot, hibát dob,
 ha a
  // hivatkozott adattag nem létezik
  private Node findIdInDoc(String id) throws Exception {
    NodeList nl1 = this.root.getChildNodes();
    for (int i = 0; i < nl1.getLength(); i++) {
      if (nl1.item(i).getNodeType() == Node.ELEMENT_NODE) {
        for (int j = 0; j < nl1.item(i).getChildNodes().getLength();
    j++) {
          if (nl1.item(i).getChildNodes().item(j).getNodeType() ==
 Node.ELEMENT_NODE) {
            for (int k = 0; k < nl1.item(i).getChildNodes().item(j).
 getAttributes().getLength(); k++) {
              if (nl1.item(i).getChildNodes().item(j).getAttributes
 ().item(k).getTextContent().equals(id)
                  && nl1.item(i).getChildNodes().item(j).
 getAttributes().item(k).getNodeName()
                      .equals("id")) {
                return nl1.item(i).getChildNodes().item(j);
              }
            }
          }
        }
      }
    }
    throw new Exception("Az id vagy referencia nem létezik");
  }

  // kilistázza a módosítható nodeokat bekéri az id-t és meghívja a
 módosítást
  public void selector() throws Exception {
    new DOMReadgpnwzt().listForModify();
```

```
    System.out.println("Kérem válassza ki a módosítandó nodokat az
    id megadásával: ");
    String choosen = this.in.nextLine();
    modify(choosen);
  }
}
```