JEGYZŐKÖNYV

Adatkezelés XML környezetben

Rakományszállító Hajók Adatbázisa

Készítette: **Pogácsás Benedek**

Neptunkód: **FM4Z3B**

Dátum: 2023.12.05

Tartalom

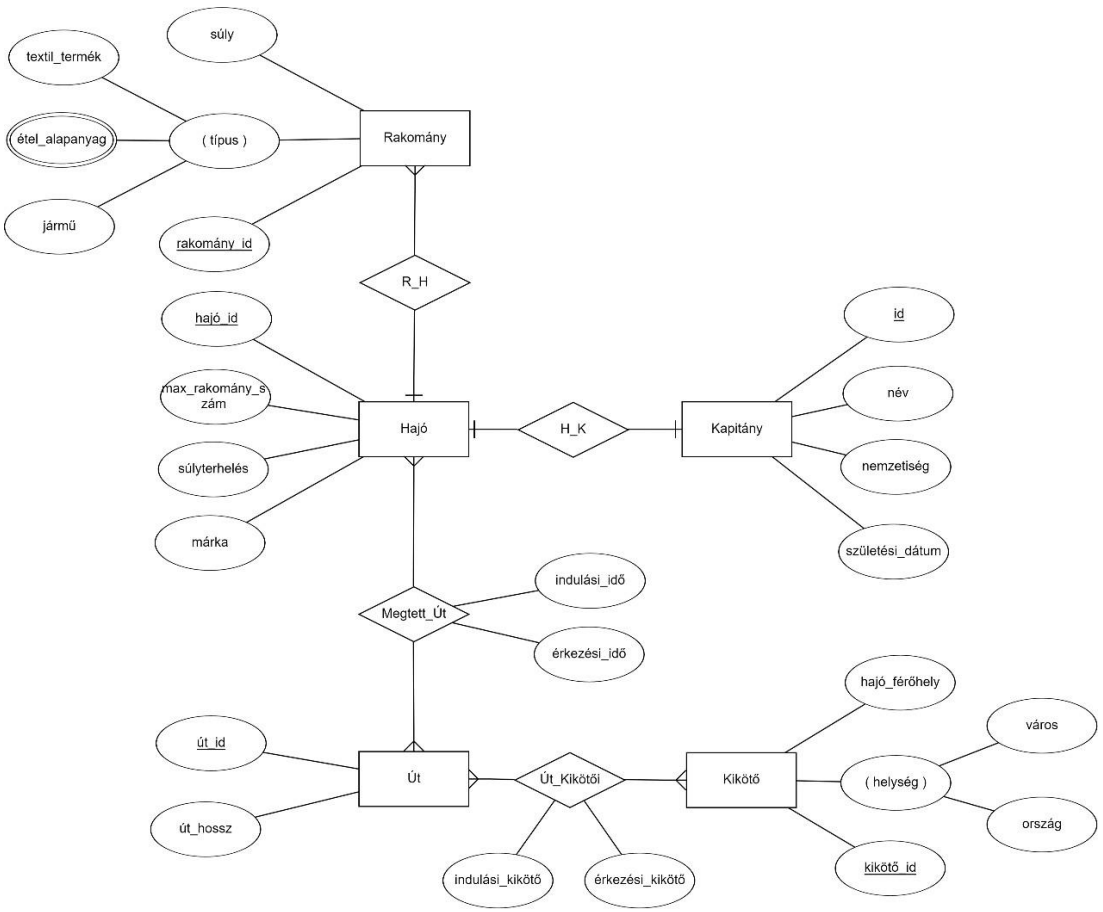
Bevezetés

Ez a jegyzőkönyv az Adatkezelés XML-ben nevű tárgy féléves feladatának dokumentálásának céljából készült.
Az általam választott téma egy rakományszállító hajók adatainak kezelése volt, amelyet az XML nyelven kellett megvalósítani. Azért választottam ezt a témát, mert személy szerint egész könnyű volt elképzelni a projekt szerkezetét, valamint egy másik tárgyból is hasonló témában csináltam egy feladatot, szóval jó alapom volt hozzá. A projekt célja, hogy egy strukturált és könnyen kezelhető adatbázist hozzon létre a hajókra, kapitányaikra, rakományokra és egyéb információkra vonatkozó adatokról.
Az XML lehetővé teszi az adatok hierarchikus és rugalmas kezelését, valamint könnyen kezelhető és átlátható formátumot biztosít. Ezen jegyzőkönyv fő célja, hogy rögzítse az adatbázis létrehozásához kapcsolódó döntéseket, folyamatokat és azokat a lépéseket, melyeket a projekt sikeres megvalósítása érdekében teszünk meg. Az adatbázis tervezése és strukturálása, az adatok gyűjtése és validálása, valamint az XML dokumentumok létrehozása és kezelése mind fontos elemek, melyekről részletesen beszámolunk a jegyzőkönyv későbbi részeiben.
Ezen felül, a jegyzőkönyv megkísérli összefoglalni a felmerülő kihívásokat és azok megoldását, valamint az esetlegesen felmerülő változtatásokat és azok hatásait az eredeti tervekre. Az átfogó dokumentáció lehetővé teszi majd számunkra, hogy nyomon kövessük a projekt fejlődését és egyértelmű képet kapjunk a folyamatokról, javítva ezzel a projektmunka hatékonyságát és átláthatóságát.

1.Feladat

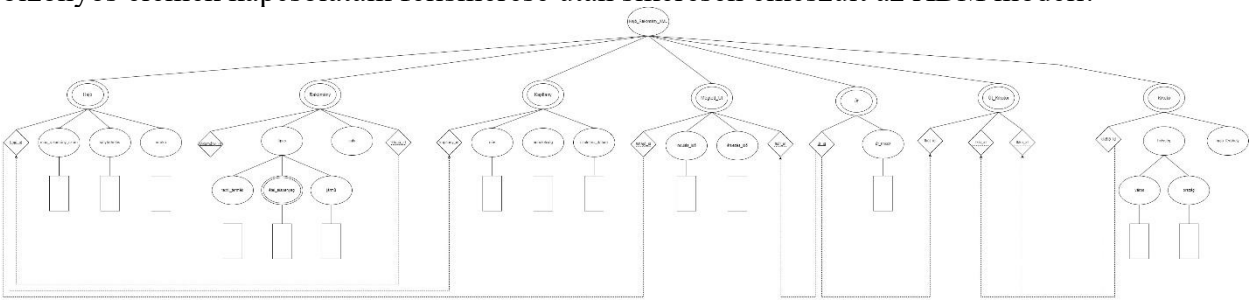
1a) Az adatbázis ER modell tervezése

A feladat ezen része nem teljesen jelentett kihívást, mivel, előző éves feladatból már megvolt az ER modell, így csak minimális változtatásokat kellett rajta elvégezni. Az ER modell:



1b) Az adatbázis konvertálása XDM modellre

Az ER modellem konvertálása XDM modellre már több erőfeszítést vett igénybe, de nem volt olyan végrehajthatatlanul nehéz feladat, szóval leültem és elkezdtem a konvertálást. A bizonyos elemek kapcsolataik felismerése után sikeresen elkészült az XDM modell:



Megjegyzés: Biztos vagyok benne, hogy a képen is látszik, de az esztétika érdeke miatt elég szélesre sikeredett ez az ábra, ezért érdemesebb magát a képet megnyitni a github repository-ból.

1c) Az XDM modell alapján XML dokumentum készítése

Mindezek után végre nekiállhattam az XML dokumentum megírására. Ezt az XDM modellnek köszönhetően egészen simán ment, így egészen hamar elkészült a kód, ami a következőféleképpen néz ki:

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

<XMLFM4Z3B xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="XMLSchemaFM4Z3B.xsd">

    <!--Hajók-->

    <Hajo hajo_id="1" kapitany="1">
        <max_rakomany_szam>4</max_rakomany_szam>
        <sulyterheles>5400</sulyterheles>
        <marka>Ever Golden</marka>
    </Hajo>

    <Hajo hajo_id="2" kapitany="2">
        <max_rakomany_szam>2</max_rakomany_szam>
        <sulyterheles>3200</sulyterheles>
        <marka>Madrid Maersk</marka>
    </Hajo>

    <Hajo hajo_id="3" kapitany="3">
        <max_rakomany_szam>6</max_rakomany_szam>
        <sulyterheles>7600</sulyterheles>
        <marka>CMA CGM</marka>
    </Hajo>

    <!--Kapitányok-->

    <Kapitany kapitany_id="1">
        <nev>Katona Ferenc</nev>
        <nemzetiseg>magyar</nemzetiseg>
        <szuletesi_datum>1974-02-14</szuletesi_datum>
    </Kapitany>

    <Kapitany kapitany_id="2">
```

```
    <nev>Charles Thompson</nev>
    <nemzetiseg>angol</nemzetiseg>
    <szuletesi_datum>1982-11-24</szuletesi_datum>
</Kapitany>

<Kapitany kapitany_id="3">
    <nev>Fernando Oliveira</nev>
    <nemzetiseg>portugál</nemzetiseg>
    <szuletesi_datum>1969-07-21</szuletesi_datum>
</Kapitany>


<!--Rakományok-->

<Rakomany rakomany_id="1" hajo="2">
    <tipus>
        <textil_termek>ruhák</textil_termek>
        <etel_alapanyag>gyümölcsök</etel_alapanyag>
        <etel_alapanyag>zöldségek</etel_alapanyag>
    </tipus>
     <suly>1400</suly>
</Rakomany>

<Rakomany rakomany_id="2" hajo="1">
    <tipus>
        <etel_alapanyag>tejtermékek</etel_alapanyag>
        <etel_alapanyag>gyümölcsök</etel_alapanyag>
        <jarmu>autók</jarmu>
    </tipus>
    <suly>3000</suly>
</Rakomany>

<Rakomany rakomany_id="3"  hajo="3">
    <tipus>
        <textil_termek>szőnyegek</textil_termek>
        <etel_alapanyag>zöldségek</etel_alapanyag>
        <etel_alapanyag>tejtermékek</etel_alapanyag>
        <jarmu>motorok</jarmu>
    </tipus>
    <suly>6400</suly>
</Rakomany>

<Rakomany rakomany_id="4" hajo="3">
    <tipus>
        <jarmu>motorcsónakok</jarmu>
    </tipus>
    <suly>1100</suly>
</Rakomany>

<!--Utak-->

<Ut ut_id="1">
    <uthossz>9841</uthossz>
```

```
</Ut>

<Ut ut_id="2">
    <uthossz>3576</uthossz>
</Ut>

<Ut ut_id="3">
    <uthossz>10500</uthossz>
</Ut>

<!--Kikötők-->

<Kikoto kikoto_id="1">
    <helyseg>
        <varos>Rio de Janeiro</varos>
        <orszag>Brazília</orszag>
    </helyseg>
    <hajo_ferohely>11</hajo_ferohely>
</Kikoto>

<Kikoto kikoto_id="2">
    <helyseg>
        <varos>Piraeus</varos>
        <orszag>Görögország</orszag>
    </helyseg>
    <hajo_ferohely>8</hajo_ferohely>
</Kikoto>

<Kikoto kikoto_id="3">
    <helyseg>
        <varos>Shanghai</varos>
        <orszag>Kína</orszag>
    </helyseg>
    <hajo_ferohely>15</hajo_ferohely>
</Kikoto>

<!--Megtett utak-->

<Megtett_Ut ut="1" hajo="1">
    <indulasi_ido>2020-11-12</indulasi_ido>
    <erkezesi_ido>2020-11-15</erkezesi_ido>
</Megtett_Ut>

<Megtett_Ut ut="2" hajo="3">
    <indulasi_ido>2020-02-26</indulasi_ido>
    <erkezesi_ido>2020-03-02</erkezesi_ido>
</Megtett_Ut>

<Megtett_Ut ut="3" hajo="2">
    <indulasi_ido>2021-04-01</indulasi_ido>
    <erkezesi_ido>2021-04-04</erkezesi_ido>
</Megtett_Ut>
```

```
    <!--Út Kikötőik-->

    <Ut_Kikotoi ut="1">
        <indulasi_kikoto>3</indulasi_kikoto>
        <erkezesi_kikoto>2</erkezesi_kikoto>
    </Ut_Kikotoi>

    <Ut_Kikotoi ut="2">
        <indulasi_kikoto>2</indulasi_kikoto>
        <erkezesi_kikoto>1</erkezesi_kikoto>
    </Ut_Kikotoi>

    <Ut_Kikotoi ut="3">
        <indulasi_kikoto>1</indulasi_kikoto>
        <erkezesi_kikoto>3</erkezesi_kikoto>
    </Ut_Kikotoi>

</XMLFM4Z3B>
```

1d) Az XML dokumentum alapján XMLSchema készítése

Az XMLSchema elkészítése során felkerült néhány probléma, amelyeket nem teljesen tudtam megoldani, de szerintem sikerült megvalósítani úgy, ahogy eredetileg akartam. A Schema kódja:

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

    <!--Egyszerű típusok-->

    <xs:element name="max_rakomany_szam" type="xs:int"/>
    <xs:element name="sulyterheles" type="xs:int"/>
    <xs:element name="marka" type="xs:string"/>
    <xs:element name="suly" type="xs:int"/>
    <xs:element name="nev" type="xs:string"/>
    <xs:element name="nemzetiseg" type="xs:string"/>
    <xs:element name="szuletesi_datum" type="xs:string"/>
    <xs:element name="indulasi_ido" type="xs:string"/>
    <xs:element name="erkezesi_ido" type="xs:string"/>
    <xs:element name="uthossz" type="xs:int"/>
    <xs:element name="indulasi_kikoto" type="xs:string"/>
    <xs:element name="erkezesi_kikoto" type="xs:string"/>
    <xs:element name="varos" type="xs:string"/>
    <xs:element name="orszag" type="xs:string"/>
    <xs:element name="hajo_ferohely" type="xs:int"/>

    <xs:element name="textil_termek" type="xs:textil_termek_tipus"/>
    <xs:element name="etel_alapanyag" type="xs:etel_alapanyag_tipus"/>
    <xs:element name="jarmu" type="xs:jarmu_tipus"/>

    <!--Saját típusok-->

    <xs:simpleType name="textil_termekTipus">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ruhák"/>
```

```
            <xs:enumeration value="szőnyekeg"/>
            <xs:enumeration value="bútor huzatok"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="etel_alapanyagTipus">
        <xs:restriction base="xs:string">
            <xs:enumeration value="zöldségek"/>
            <xs:enumeration value="gyümölcsök"/>
            <xs:enumeration value="tejtermékek"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="jarmuTipus">
        <xs:restriction base="xs:string">
            <xs:enumeration value="autók"/>
            <xs:enumeration value="motorok"/>
            <xs:enumeration value="motorcsónakok"/>
        </xs:restriction>
    </xs:simpleType>


    <!--Komplex típusok-->

    <xs:complexType name="hajoTipus">
        <xs:sequence>
            <xs:element ref="max_rakomany_szam" maxOccurs="1"/>
            <xs:element ref="sulyterheles" maxOccurs="1"/>
            <xs:element ref="marka" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="hajo_id" type="xs:int" use="required"/>
    </xs:complexType>

    <xs:complexType name="kapitanyTipus">
        <xs:sequence>
            <xs:element ref="nev" maxOccurs="1"/>
            <xs:element ref="nemzetiseg" maxOccurs="1"/>
            <xs:element ref="szuletesi_datum" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="kapitany_id" type="xs:int" use="required"/>
    </xs:complexType>

    <xs:complexType name="rakomanyTipus">
        <xs:sequence>
            <xs:element name="tipus"/>
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="textil_termek" maxOccurs="1"/>
                        <xs:element ref="etel_alapanyag" maxOccurs="unbounded"/>
                        <xs:element ref="jarmu" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
```

```
            <xs:element ref="suly" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="rakomany_id" type="xs:int" use="required"/>
        <xs:attribute name="hajo" type="xs:int" use="required"/>
    </xs:complexType>

    <xs:complexType name="utTipus">
        <xs:sequence>
            <xs:element ref="uthossz" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="ut_id" type="xs:int" use="required"/>
    </xs:complexType>

    <xs:complexType name="kikotoTipus">
        <xs:sequence>
            <xs:element name="tipus"/>
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="varos" maxOccurs="1"/>
                        <xs:element ref="orszag" maxOccurs="1"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element ref="hajo_ferohely" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="kikoto_id" type="xs:int" use="required"/>
    </xs:complexType>

    <xs:complexType name="megtett_utTipus">
        <xs:sequence>
            <xs:element ref="indulasi_ido" maxOccurs="1"/>
            <xs:element ref="erkezesi_ido" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="ut_id" type="xs:int" use="required"/>
        <xs:attribute name="hajo_id" type="xs:int" use="required"/>
    </xs:complexType>

    <xs:complexType name="ut_kikotoiTipus">
        <xs:sequence>
            <xs:element ref="indulasi_kikoto" maxOccurs="1"/>
            <xs:element ref="erkezesi_kikoto" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="ut_id" type="xs:int" use="required"/>
    </xs:complexType>

    <!--Gyökérelem elemei-->

    <xs:element name="Hajo_Rakomany_XML">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Hajo" type="hajoTipus" minOccurs="1"
maxOccurs="unbounded"/>
```

```
                <xs:element name="Kapitany" type="kapitanyTipus" minOccurs="1"
maxOccurs="unbounded"/>
                <xs:element name="Rakomany" type="rakomanyTipus" minOccurs="1"
maxOccurs="unbounded"/>
                <xs:element name="Ut" type="utTipus" minOccurs="1"
maxOccurs="unbounded"/>
                <xs:element name="Kikoto" type="kikotoTipus" minOccurs="1"
maxOccurs="unbounded"/>
                <xs:element name="Megtett_Ut" type="megtett_ut" minOccurs="1"
maxOccurs="unbounded"/>
                <xs:element name="Ut_Kikotoi" type="ut_kikotoi" minOccurs="1"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:sequence>

    <!--Elsődleges kulcsok-->

        <xs:key name="Hajo_kod">
            <xs:selector xpath="Hajo"/>
            <xs:field xpath="@hajo_kod"/>
        </xs:key>

        <xs:key name="Kapitany_kod">
            <xs:selector xpath="Kapitany"/>
            <xs:field xpath="@kapitany_kod"/>
        </xs:key>

        <xs:key name="Rakomany_kod">
            <xs:selector xpath="Rakomany"/>
            <xs:field xpath="@rakomany_kod"/>
        </xs:key>

        <xs:key name="Ut_kod">
            <xs:selector xpath="Ut"/>
            <xs:field xpath="@ut_kod"/>
        </xs:key>

        <xs:key name="Kikoto_kod">
            <xs:selector xpath="Kikoto"/>
            <xs:field xpath="@kikoto_kod"/>
        </xs:key>

    <!--Idegen kulcsok-->

        <xs:keyref name="R_H_kod" refer="Hajo_kod">
            <xs:selector xpath="Rakomany"/>
            <xs:field xpath="@hajo"/>
        </xs:keyref>

        <xs:keyref name="MU_H_kod" refer="Hajo_kod">
            <xs:selector xpath="Megtett_Ut"/>
            <xs:field xpath="@hajo"/>
```

```
        </xs:keyref>

        <xs:keyref name="MU_U_kod" refer="Ut_kod">
            <xs:selector xpath="Megtett_Ut"/>
            <xs:field xpath="@ut"/>
        </xs:keyref>

        <xs:keyref name="UK_U_kod" refer="Ut_kod">
            <xs:selector xpath="Ut_Kikotoi"/>
            <xs:field xpath="@fkut_id"/>
        </xs:keyref>

        <xs:keyref name="UK_KI_kod" refer="Kikoto_id">
            <xs:selector xpath="Ut_Kikotoi"/>
            <xs:field xpath="indulasi_kikoto"/>
        </xs:keyref>

        <xs:keyref name="UK_KE_kod" refer="Kikoto_id">
            <xs:selector xpath="Ut_Kikotoi"/>
            <xs:field xpath="erkezesi_kikoto"/>
        </xs:keyref>

    <!--1:1 Kapcsolat-->

        <xs:keyref name="Hajo_Kapitanya">
            <xs:selector xpath="Hajo"/>
            <xs:field xpath="@kapitany"/>
        </xs:keyref>

</xs:schema>
```

Megjegyzés: Néhány helyen nem fért ki teljesen a kód, így elcsúszott.

2.Feladat

Az XML fálj kódja megtalálható az 1c) pontban, és mivel elég hosszú, nem szeretnék több oldalt erre használni.

Ehhez a feladathoz egy DOM programot kellett készíteni, amivel különféle parancsokat és műveleteket hajthatunk végre az XML fájlunkon. Hogy könnyebben menjen a tesztelés, készítettem egy kis konzolos menüt, ami egy külön Main classba került:

```
package hu.domparse.fm4z3b;

import java.util.Scanner;

public class Main {

	public static void main(String[] args) {

		String url = "XMLFM4Z3B.xml";

		int input;

		System.out.println("Welcome!");

		Scanner scanner = new Scanner(System.in);
		input = 0;
		System.out.println("All the available commands:");
		System.out.println("  1. Read its content.");
		System.out.println("  2. Read its content in a structured manner.");
```

```
            System.out.println("  3. Modify an already existing element.");
            System.out.println("  4. Write the file in a tree-like structure.");
            System.out.println("\n  Queries:");
            System.out.println("  5. Specific ID search.");
            System.out.println("  6. Which ship can carry the most weight?");
            System.out.println("  7. Which road is the shortest?");
            System.out.println("  8. Info about each roads Docs.");
            System.out.println("  9. Can the ship carry the cargo?");
            System.out.println("\nSelect what you want to do in XMLFM4Z3B.xml");
            input = scanner.nextInt();

            switch(input) {
                  case 1: DomReadFM4Z3B.Read(url); break;
                  case 2: DomReadFM4Z3B.StructuredRead(url); break;
                  case 3: DomModifyFM4Z3B.Modify(url);
DomReadFM4Z3B.StructuredRead(url); break;
                  case 4:      DomWriteFM4Z3B.Write(); break;
                  case 5: DomQueryFM4Z3B.SpecificIdInfo(url); break;
                  case 6:      DomQueryFM4Z3B.ShipCargoWeightOrdered(url); break;
                  case 7:      DomQueryFM4Z3B.FindShortestRoad(url); break;
                  case 8: DomQueryFM4Z3B.RoadDocsInfo(url); break;
                  case 9: DomQueryFM4Z3B.CanItCarry(url); break;
                  default: System.out.println("No such command.");
            }
            scanner.close();
      }
}
```

2a) Adatolvasás

Az adatolvasáshoz két külön olvasást készítettem, amelyek másképpen működnek. Ezeken kívül még néhány segédmetódust csináltam, hogy jobban átláthatóbb legyen a kód, ami végül így sikerült:

```
package hu.domparse.fm4z3b;

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

import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.NamedNodeMap;

public class DomReadFM4Z3B {

      //Sima kiíratás
      public static void Read(String url) {
            try {
          File inputFile = new File(url);
          DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
          DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
          Document doc = dBuilder.parse(inputFile);
          doc.getDocumentElement().normalize();

          traverseNodes(doc.getDocumentElement(), "");

        } catch (Exception e) {
          e.printStackTrace();
        }
      }

      //Struktúrált kiíratás
      public static void StructuredRead(String url) {
```

```
            try {
            File inputFile = new File(url);
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            String xmlContent = docToString(doc);
            System.out.println(xmlContent);

        } catch (Exception e) {
            e.printStackTrace();
        }
         }

         //Documentből Stringre konvertelő függvény
         private static String docToString(Document doc) {
        try {
            javax.xml.transform.TransformerFactory tf =
javax.xml.transform.TransformerFactory.newInstance();
            javax.xml.transform.Transformer transformer = tf.newTransformer();
           transformer.setOutputProperty(javax.xml.transform.OutputKeys.OMIT_XML_DECLARATION,
"no");
            transformer.setOutputProperty(javax.xml.transform.OutputKeys.METHOD, "xml");
            transformer.setOutputProperty(javax.xml.transform.OutputKeys.INDENT, "yes");
            transformer.setOutputProperty(javax.xml.transform.OutputKeys.ENCODING, "UTF-8");
            java.io.StringWriter writer = new java.io.StringWriter();
            transformer.transform(new javax.xml.transform.dom.DOMSource(doc), new
javax.xml.transform.stream.StreamResult(writer));
            return writer.getBuffer().toString();
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

         //Rekurzív metódus, ami képes kiírni egy elem gyerek elemét
         private static void traverseNodes(Node node, String indent) {
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            String elementName = node.getNodeName();
            System.out.print(indent + elementName + ":");

            if (node.hasAttributes()) {
                NamedNodeMap attributes = node.getAttributes();
                System.out.print(" (");
                for (int i = 0; i < attributes.getLength(); i++) {
                    Node attribute = attributes.item(i);
                    if (i == 0) {
                      System.out.print(attribute.getNodeName() + " = " + attribute.getNodeValue());
                    } else {
                      System.out.print(", " + attribute.getNodeName() + " = " + attribute.getNodeValue());
                    }
                }
                System.out.print(")");
            }

            if (node.hasChildNodes()) {
                NodeList childNodes = node.getChildNodes();
                for (int j = 0; j < childNodes.getLength(); j++) {
```

```
                Node childNode = childNodes.item(j);
                if (childNode.getNodeType() == Node.ELEMENT_NODE) {
                    System.out.print("\n");
                  traverseNodes(childNode, indent + "\t");
                } else if (childNode.getNodeType() == Node.TEXT_NODE &&
!childNode.getNodeValue().trim().isEmpty()) {
                    System.out.print(" " + childNode.getNodeValue().trim());
                }
            }
        }
    }
  }
}
```

2b) Adatmódosítás

Az adatmódosításhozegyetlen metódust készítettem, amely a bekért információ alapján gyakorlatilag bármit képes átírni az XML fájlban.

Itt az 5 változtatás:

1.

```
Enter the parent element's name: Hajo
Enter hajo_id's value: 2
Enter the target element's name: marka
Enter the target element's new value: Madrid Maersk
XML file updated successfully.
```

2.

```
Enter the parent element's name: Kapitany
Enter kapitany_id's value: 2
Enter the target element's name: nemzetiseg
Enter the target element's new value: amerikai
XML file updated successfully.
```

3.

```
Enter the parent element's name: Rakomany
Enter rakomany_id's value: 4
Enter the target element's name: suly
Enter the target element's new value: 1300
XML file updated successfully.
```

4.

```
Enter the parent element's name: Kikoto
Enter kikoto_id's value: 3
Enter the target element's name: orszag
Enter the target element's new value: Japan
XML file updated successfully.
```

5.

```
Enter the parent element's name: Kikoto
Enter kikoto_id's value: 3
Enter the target element's name: varos
Enter the target element's new value: Tokyo
XML file updated successfully.
```

Valamint itt az osztály kódja:

```
package hu.domparse.fm4z3b;

import java.io.File;
import java.util.Scanner;
import javax.xml.parsers.DocumentBuilderFactory;
```

```java
import javax.xml.parsers.DocumentBuilder;

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

public class DomModifyFM4Z3B {

	//Egy metódus, amely megadott információk után módosítja az XML fájl adatait
	public static void Modify(String url) {
		 try {
		File inputFile = new File(url);
		DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
		DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
		Document doc = dBuilder.parse(inputFile);
		doc.getDocumentElement().normalize();

		Scanner scanner = new Scanner(System.in);
		String parentElementName;
		String parentAttributeName = "", parentAttributeName2 = "";
		String parentAttributeValue, parentAttributeValue2 = "";
		String targetElementName;
		NodeList list;
		boolean ok = false;

		do {
		    System.out.print("Enter the parent element's name: ");
		    parentElementName = scanner.nextLine();
		    list = doc.getElementsByTagName(parentElementName);

		    if(list.getLength() == 0) {
		        System.out.println("No such parent element found.");
		    }
		} while(list.getLength() == 0);

		switch(parentElementName) {
		 case "Hajo": parentAttributeName = "hajo_id"; break;
		 case "Kapitany": parentAttributeName = "kapitany_id"; break;
		 case "Rakomany": parentAttributeName = "rakomany_id"; break;
		 case "Ut": parentAttributeName = "ut_id"; break;
		 case "Kikoto": parentAttributeName = "kikoto_id"; break;
		}

		if (parentElementName.equals("Megtett_Ut")) {
		 System.out.print("Enter hajo's value: ");
		    parentAttributeValue = scanner.nextLine();
		    parentAttributeName = "hajo";

		    System.out.print("Enter ut's value: ");
		    parentAttributeValue2 = scanner.nextLine();
		    parentAttributeName2 = "ut";
		} else {
		 System.out.print("Enter " + parentAttributeName + "'s value: ");
		    parentAttributeValue = scanner.nextLine();
		}

		do {
		 System.out.print("Enter the target element's name: ");
		    targetElementName = scanner.nextLine();
		    list = doc.getElementsByTagName(targetElementName);

		    if(list.getLength() == 0) {
		        System.out.println("No such target element found.");
		    }
```

```
            } while(list.getLength() == 0);

            System.out.print("Enter the target element's new value: ");
            String targetElementNewValue = scanner.nextLine();

            NodeList parentList = doc.getElementsByTagName(parentElementName);
            for (int i = 0; i < parentList.getLength(); i++) {
                Element parentElement = (Element) parentList.item(i);
                if (parentElementName.equals("Megtett_Ut")) {
                    if (parentElement.hasAttribute(parentAttributeName) &&
parentElement.getAttribute(parentAttributeName).equals(parentAttributeValue) &&
                                parentElement.hasAttribute(parentAttributeName2) &&
parentElement.getAttribute(parentAttributeName2).equals(parentAttributeValue2)) {
                            ok = true;
                            NodeList childList =
parentElement.getElementsByTagName(targetElementName);
                        for (int j = 0; j < childList.getLength(); j++) {
                            Element targetElement = (Element) childList.item(j);
                            targetElement.setTextContent(targetElementNewValue);
                        }
                    }

                } else {
                    if (parentElement.hasAttribute(parentAttributeName) &&
parentElement.getAttribute(parentAttributeName).equals(parentAttributeValue)) {
                        ok = true;
                          NodeList childList =
parentElement.getElementsByTagName(targetElementName);
                        for (int j = 0; j < childList.getLength(); j++) {
                            Element targetElement = (Element) childList.item(j);
                            targetElement.setTextContent(targetElementNewValue);
                        }
                    }
                }
            }

            javax.xml.transform.TransformerFactory transformerFactory =
javax.xml.transform.TransformerFactory.newInstance();
            javax.xml.transform.Transformer transformer =
transformerFactory.newTransformer();
            javax.xml.transform.dom.DOMSource source = new
javax.xml.transform.dom.DOMSource(doc);
            javax.xml.transform.stream.StreamResult result = new
javax.xml.transform.stream.StreamResult(new File(url));
            transformer.transform(source, result);

            if (ok) {
              System.out.println("XML file updated successfully.");
            } else {
              System.out.println("Update failed.");
            }

            scanner.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
       }
}
```

2c) Adatlekérdezés

Lekérdezésből 5 darabot csináltam, amelyek a következőek:

1. Element és id megadása után kiírja az információt

```
public static void SpecificIdInfo(String url) {
```

```
            boolean check = false;
            try {
            File inputFile = new File(url);
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            String elementName, attributeValue, attributeName = "";
            Scanner scanner = new Scanner(System.in);
            boolean echeck;

            do {
                echeck = true;
             System.out.println("Which elements information do you want to see?");
                elementName = scanner.nextLine();

                switch(elementName) {
                    case "Hajo": attributeName = "hajo_id"; break;
                    case "Kapitany": attributeName = "kapitany_id"; break;
                    case "Rakomany": attributeName = "rakomany_id"; break;
                    case "Ut": attributeName = "ut_id"; break;
                    case "Kikoto": attributeName = "kikoto_id"; break;
                    default: System.out.println("No such element!"); echeck = false;
                }
            } while(!echeck);

            System.out.println("Which "+ elementName + " information do you want to
see?");
            attributeValue = scanner.nextLine();

            scanner.close();

            NodeList elements = doc.getElementsByTagName("*");
            for (int i = 0; i < elements.getLength(); i++) {
                Element element = (Element) elements.item(i);
                if (element.hasAttribute(attributeName) &&
element.getAttribute(attributeName).equals(attributeValue)) {
                    check = true;
                    System.out.println(element.getNodeName() + ": ");
                    NodeList childNodes = element.getChildNodes();
                    for (int j = 0; j < childNodes.getLength(); j++) {
                        if (childNodes.item(j).getNodeType() ==
org.w3c.dom.Node.ELEMENT_NODE) {
                            System.out.println("\t" +
childNodes.item(j).getNodeName() + ": " +
childNodes.item(j).getTextContent().trim());
                        }
                    }
                    System.out.println();
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        if (!check) {
             System.out.println("No element with this id.");
        }
       }
```

2. Hajók rendezése súlyterhelés alapján(csökkenő)

```
public static void ShipCargoWeightOrdered(String url) {
        try {
            Map<String, Double> weightMap = new HashMap<>();
```

```
            File inputFile = new File(url);
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            NodeList shipList = doc.getElementsByTagName("Hajo");
            for (int i = 0; i < shipList.getLength(); i++) {
                Element ship = (Element) shipList.item(i);
                String shipId = ship.getAttribute("hajo_id");
                NodeList weightList = ship.getElementsByTagName("sulyterheles");
                if (weightList.getLength() > 0) {
                    Element weightElement = (Element) weightList.item(0);
                    double weight =
Double.parseDouble(weightElement.getTextContent().trim());
                    weightMap.put(shipId, weight);
                }
            }

            List<Map.Entry<String, Double>> sortedWeights = new
ArrayList<>(weightMap.entrySet());
            sortedWeights.sort((entry1, entry2) -> Double.compare(entry2.getValue(),
entry1.getValue()));

            System.out.println("Weight - Ship ID");
            for (Map.Entry<String, Double> entry : sortedWeights) {
                System.out.println(entry.getValue() + " - " + entry.getKey());
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

3. Legrövidebb út megkeresése

```
public static void FindShortestRoad(String url) {
        try {
            int shortestRoadLength = Integer.MAX_VALUE;
            String shortestRoadId = "";

            File inputFile = new File(url);
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            NodeList roadList = doc.getElementsByTagName("Ut");
            for (int i = 0; i < roadList.getLength(); i++) {
                Element road = (Element) roadList.item(i);
                String roadId = road.getAttribute("ut_id");
                Element roadLengthElement = (Element)
road.getElementsByTagName("uthossz").item(0);
                int roadLength =
Integer.parseInt(roadLengthElement.getTextContent().trim());
                if (roadLength < shortestRoadLength) {
                    shortestRoadLength = roadLength;
                    shortestRoadId = roadId;
                }
            }

            if (!shortestRoadId.isEmpty()) {
                System.out.println("Shortest Road ID: " + shortestRoadId);
```

```
                System.out.println("Shortest Road Length: " + shortestRoadLength + "
km");
            } else {
                System.out.println("No road found.");
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

4. Utak kezdő és vég kikötőinek információinak kiírása

```
public static void RoadDocsInfo(String url) {
        try {
            File inputFile = new File(url);
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            NodeList roadDocsList = doc.getElementsByTagName("Ut_Kikotoi");
            for (int i = 0; i < roadDocsList.getLength(); i++) {
                Element roadDocs = (Element) roadDocsList.item(i);
                String roadAttribute = roadDocs.getAttribute("ut");

                Element startDoc = (Element)
roadDocs.getElementsByTagName("indulasi_kikoto").item(0);
                Element endDoc = (Element)
roadDocs.getElementsByTagName("erkezesi_kikoto").item(0);
                String startDocId = startDoc.getTextContent().trim();
                String endDocId = endDoc.getTextContent().trim();

                System.out.println("Road ID: " + roadAttribute);
                System.out.println("\tFrom: ");
                outputDocInfo(doc, startDocId);
                System.out.println("\tTo: ");
                outputDocInfo(doc, endDocId);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

      //Segéd metódus a kikötők kiírásához
      private static void outputDocInfo(Document doc, String docId) {
        NodeList docList = doc.getElementsByTagName("Kikoto");
        for (int j = 0; j < docList.getLength(); j++) {
            Element docElement = (Element) docList.item(j);
            String docIdAttribute = docElement.getAttribute("kikoto_id");
            if (docId.equals(docIdAttribute)) {
                Element place = (Element)
docElement.getElementsByTagName("helyseg").item(0);
                Element shipStorage = (Element)
docElement.getElementsByTagName("hajo_ferohely").item(0);

                Element country = (Element)
place.getElementsByTagName("orszag").item(0);
                Element city = (Element) place.getElementsByTagName("varos").item(0);
                String countryName = country.getTextContent().trim();
                String cityName = city.getTextContent().trim();

                String shipStorageValue = shipStorage.getTextContent().trim();

                System.out.println("\t\tCountry: " + countryName);
                System.out.println("\t\tCity: " + cityName);
```

```
                System.out.println("\t\tShip Storage: " + shipStorageValue);
                System.out.println();
            }
        }
    }
```

5. Megnézi hogy egy bekérd rakományt eltudna-e szállítani egy bekért hajó

```
public static void CanItCarry(String url) {
             try {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter rakomany_id: ");
            String cargoId = scanner.nextLine().trim();
            System.out.print("Enter hajo_id: ");
            String shipId = scanner.nextLine().trim();

            File inputFile = new File(url);
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            double cargoWeight = 0.0;
            double shipWeightLimit = 0.0;

            NodeList cargoList = doc.getElementsByTagName("Rakomany");
            for (int i = 0; i < cargoList.getLength(); i++) {
                Element cargo = (Element) cargoList.item(i);
                String cargoID = cargo.getAttribute("rakomany_id");
                if (cargoID.equals(cargoId)) {
                    Element weight = (Element)
cargo.getElementsByTagName("suly").item(0);
                    cargoWeight = Double.parseDouble(weight.getTextContent().trim());
                    break;
                }
            }

            NodeList shipList = doc.getElementsByTagName("Hajo");
            for (int i = 0; i < shipList.getLength(); i++) {
                Element ship = (Element) shipList.item(i);
                String shipID = ship.getAttribute("hajo_id");
                if (shipID.equals(shipId)) {
                    Element weightLimit = (Element)
ship.getElementsByTagName("sulyterheles").item(0);
                    shipWeightLimit =
Double.parseDouble(weightLimit.getTextContent().trim());
                    break;
                }
            }

            if (cargoWeight > shipWeightLimit) {
                System.out.println("The ship WOULD NOT be able to carry this
cargo.");
            } else {
                System.out.println("The ship WOULD be able to carry this cargo.");
            }

            scanner.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
       }
```

2d) Adatkiírás

Az adatok kiírása konzolra és egy fájlba volt az utolsó feladat. Ezt a következőféleképpen valósítottam

meg:

```
package hu.domparse.fm4z3b;

import java.io.File;
import java.io.FileWriter;
import java.io.PrintWriter;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;

import java.util.StringJoiner;

import org.w3c.dom.*;

public class DomWriteFM4Z3B {

	public static void Write() {
		 try {
			DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
			DocumentBuilder builder = dbf.newDocumentBuilder();
			Document document = builder.newDocument();

			//Gyökérelem
			Element rootElement = document.createElement("XMLFM4Z3B");
			rootElement.setAttribute("xmlns:xsi", "http://www.w3.org/2001/XMLSchema-
instance");
			rootElement.setAttribute("xsi:noNamespaceSchemaLocation",
"XMLSchemaFM4Z3B.xsd");
			document.appendChild(rootElement);

			//Hajok
			createHajo(document, rootElement, "1", "1", "4", "5400", "Ever Golden");
			createHajo(document, rootElement, "2", "2", "2", "3200", "Madrid
Maersk");
			createHajo(document, rootElement, "3", "3", "6", "7600", "CMA CGM");

			//Kapitanyok
			createKapitany(document, rootElement, "1", "Katona Ferenc", "magyar",
"1974-02-14");
			createKapitany(document, rootElement, "2", "Charles Thompson", "angol",
"1982-11-24");
			createKapitany(document, rootElement, "3", "Fernando Oliveira",
"portugál", "1969-07-21");

			//Rakomanyok
			String[] ea1 = {"gyümölcsök", "zöldségek"};
			createRakomany(document, rootElement, "1", "2", ea1, "nincs", "ruhák",
"1400");
			String[] ea2 = {"tejtermékek", "gyümölcsök"};
			createRakomany(document, rootElement, "2", "1", ea2, "autók", "nincs",
"3000");
			String[] ea3 = {"zöldségek", "tejtermékek"};
			createRakomany(document, rootElement, "3", "3", ea3, "motorok",
"szőnyegek", "6400");
			String[] ea4 = {"nincs"};
			createRakomany(document, rootElement, "4", "3", ea4, "autók", "nincs",
"3000");

			//Utak
			createUt(document, rootElement, "1", "9841");
			createUt(document, rootElement, "2", "3576");
```

```
            createUt(document, rootElement, "3", "10500");

            //Kikotok
            createKikoto(document, rootElement, "1", "Rio de Janeiro", "Brazília",
"11");
            createKikoto(document, rootElement, "2", "Piraeus", "Görögország", "8");
            createKikoto(document, rootElement, "3", "Shanghai", "Kína", "15");

            //Megtett_Utak
            createMegtettUt(document, rootElement, "1", "1", "2020-11-12", "2020-11-
15");
            createMegtettUt(document, rootElement, "2", "3", "2020-02-26", "2020-03-
02");
            createMegtettUt(document, rootElement, "3", "2", "2021-04-01", "2021-04-
04");

            //Utak_Kikotoik
            createUtKikotoi(document, rootElement, "1", "3", "2");
            createUtKikotoi(document, rootElement, "2", "2", "1");
            createUtKikotoi(document, rootElement, "3", "1", "3");

            //Dokumentum kiírása a koznolra és egy fájlba
            TransformerFactory transformerFactory = TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
            transformer.setOutputProperty(OutputKeys.INDENT, "yes");
            transformer.setOutputProperty("{https://xml.apache.org/xslt}indent-
amount", "4");

            printDoc(document);
             } catch (Exception e) {
            e.printStackTrace();
        }
      }

      //Hajo hozzáadása
      private static void createHajo(Document doc, Element root, String hajoid,
String kapitany, String mrsz, String st, String mk) {
        Element hajo = doc.createElement("Hajo");
        hajo.setAttribute("hajo_id", hajoid);
        hajo.setAttribute("kapitany", kapitany);

        Element Element_mrsz = createElement(doc, "max_rakomany_szam", mrsz);
        Element Element_st = createElement(doc, "sulyterheles", st);
        Element Element_mk = createElement(doc, "marka", mk);
        hajo.appendChild(Element_mrsz);
        hajo.appendChild(Element_st);
        hajo.appendChild(Element_mk);

        root.appendChild(hajo);
      }

      //Kapitany hozzáadása
      private static void createKapitany(Document doc, Element root, String
kapitanyid, String nev, String nzt, String szuld) {
        Element kapitany = doc.createElement("Kapitany");
        kapitany.setAttribute("kapitany_id", kapitanyid);

        Element Element_nev = createElement(doc, "nev", nev);
        Element Element_nzt = createElement(doc, "nemzetiseg", nzt);
        Element Element_szuld = createElement(doc, "szuletesi_datum", szuld);
        kapitany.appendChild(Element_nev);
        kapitany.appendChild(Element_nzt);
        kapitany.appendChild(Element_szuld);
```

```
        root.appendChild(kapitany);
    }

    //Rakomany hozzáadása
    private static void createRakomany(Document doc, Element root, String
rakomanyid, String hajoid,String[] ea, String jrm, String tt, String suly) {
        Element rakomany = doc.createElement("Rakomany");
        rakomany.setAttribute("rakomanyid", rakomanyid);
        rakomany.setAttribute("hajo", hajoid);
        Element tipus = doc.createElement("tipus");
        rakomany.appendChild(tipus);

        Element[] Element_ea = new Element[ea.length];
        if (ea.length >= 1 && ea[0].equals("nincs") == false) {
            for (int i = 0; i < ea.length; i++) {
                Element_ea[i] = createElement(doc, "etel_alapanyag", ea[i]);
                tipus.appendChild(Element_ea[i]);
            }
        }
        if (jrm.equals("nincs") == false) {
            Element Element_jrm = createElement(doc, "jarmu", jrm);
            tipus.appendChild(Element_jrm);
        }
        if (tt.equals("nincs") == false) {
            Element Element_tt = createElement(doc, "textil_termek", tt);
            tipus.appendChild(Element_tt);
        }

        Element Element_suly = createElement(doc, "suly", suly);
        rakomany.appendChild(Element_suly);

        root.appendChild(rakomany);
    }

    //Ut hozzáadása
    private static void createUt(Document doc, Element root, String utid, String
uh) {
        Element ut = doc.createElement("Ut");
        ut.setAttribute("ut_id", utid);

        Element Element_uh = createElement(doc, "uthossz", uh);
        ut.appendChild(Element_uh);

        root.appendChild(ut);
    }

    //Kikoto hozzáadása
    private static void createKikoto(Document doc, Element root, String kikotoid,
String og, String vs, String hf) {
        Element kikoto = doc.createElement("Kikoto");
        kikoto.setAttribute("kikoto_id", kikotoid);

        Element hely = doc.createElement("helyseg");

        Element Element_og = createElement(doc, "orszag", og);
        Element Element_vs = createElement(doc, "varos", vs);
        Element Element_hf = createElement(doc, "hajo_ferohely", hf);
        hely.appendChild(Element_og);
        hely.appendChild(Element_vs);
        kikoto.appendChild(hely);
        kikoto.appendChild(Element_hf);

        root.appendChild(kikoto);
```

```
	}

	//Megtett_Ut hozzáadása
	private static void createMegtettUt(Document doc, Element root, String utid,
String hajoid, String ii, String ei) {
	 Element mu = doc.createElement("Megtett_Ut");
	 mu.setAttribute("ut_id", utid);
	 mu.setAttribute("hajo", hajoid);

	 Element Element_ii = createElement(doc, "indulasi_ido", ii);
	 Element Element_ei = createElement(doc, "erkezesi_ido", ei);
	 mu.appendChild(Element_ii);
	 mu.appendChild(Element_ei);

	 root.appendChild(mu);
	}

	//Ut_Kikotoi hozzáadása
	private static void createUtKikotoi(Document doc, Element root, String utid,
String ik, String ek) {
		Element mu = doc.createElement("Ut_Kikotoi");
		mu.setAttribute("ut_id", utid);

		Element Element_ik = createElement(doc, "indulasi_kikoto", ik);
		Element Element_ek = createElement(doc, "erkezesi_kikoto", ek);
		mu.appendChild(Element_ik);
		mu.appendChild(Element_ek);

		root.appendChild(mu);
	}

	//Segédmetódus Element-ek készítéséhez
	private static Element createElement(Document document, String name, String
value) {
		Element element = document.createElement(name);
		element.appendChild(document.createTextNode(value));

		return element;
	}

	//Segédmetódus a NodeList-ek kezeléséhez
	private static void printNodes(NodeList nodeList, PrintWriter writer) {
		for (int i = 0; i < nodeList.getLength(); i++) {
			Node node = nodeList.item(i);
			printNode(node, 1, writer);
			System.out.println("");
			writer.println("");
		}
	}

	//Segédmetódus a Node-ok kezeléséhez
	private static void printNode(Node node, int indent, PrintWriter writer) {
		if (node.getNodeType() == Node.ELEMENT_NODE) {
			Element element = (Element) node;
			String nodeName = element.getTagName();
			StringJoiner attributes = new StringJoiner(" ");
			NamedNodeMap attributeMap = element.getAttributes();
			//Elem neve és attribútumainak kiírása
			for (int i = 0; i < attributeMap.getLength(); i++) {
				Node attribute = attributeMap.item(i);
				attributes.add(attribute.getNodeName() + "=\"" +
attribute.getNodeValue() + "\"");
			}
```

```
                    System.out.print(indentHandler(indent));
                    System.out.print("<" + nodeName + " " + attributes.toString() +
">");

                    writer.print(indentHandler(indent));
                    writer.print("<" + nodeName + " " + attributes.toString() + ">");

                    NodeList children = element.getChildNodes();
                    if (children.getLength() == 1 && children.item(0).getNodeType()
== Node.TEXT_NODE) {
                           System.out.print(children.item(0).getNodeValue());
                           writer.print(children.item(0).getNodeValue());
                    } else {
                           System.out.println();
                           writer.println();
                           for (int i = 0; i < children.getLength(); i++) {
                                  printNode(children.item(i), indent + 1, writer);
                           }
                           System.out.print(indentHandler(indent));
                           writer.print(indentHandler(indent));
                    }
                    System.out.println("</" + nodeName + ">");
                    writer.println("</" + nodeName + ">");
             }
      }

      private static void printDoc(Document doc) {
             try {
                    File url = new File("XMLFM4Z3B1.xml");
                    PrintWriter writer = new PrintWriter(new FileWriter(url, true));

                    Element rootElement = doc.getDocumentElement();
                    String rootName = rootElement.getTagName();

                    //Gyökérelem attribútumainak kiírása
                    StringJoiner rootAttributes = new StringJoiner(" ");

                    //Gyökérelem attribútumainak lekérése
                    NamedNodeMap rootAttributeMap = rootElement.getAttributes();

                    for (int i = 0; i < rootAttributeMap.getLength(); i++) {
                           Node attribute = rootAttributeMap.item(i);
                           rootAttributes.add(attribute.getNodeName() + "=\"" +
attribute.getNodeValue() + "\"");
                    }

                    System.out.print("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
                    writer.print("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");

                    System.out.print("<" + rootName + " " + rootAttributes.toString()
+ ">\n");
                    writer.print("<" + rootName + " " + rootAttributes.toString() +
">\n");

                    NodeList list_of_Hajo = doc.getElementsByTagName("Hajo");
                    NodeList list_of_Kapitany = doc.getElementsByTagName("Kapitany");
                    NodeList list_of_Rakomany = doc.getElementsByTagName("Rakomany");
                    NodeList list_of_Ut = doc.getElementsByTagName("Ut");
                    NodeList list_of_Kikoto = doc.getElementsByTagName("Kikoto");
                    NodeList list_of_Megtett_Ut =
doc.getElementsByTagName("Megtett_Ut");
                    NodeList list_of_Ut_Kikotoi =
doc.getElementsByTagName("Ut_Kikotoi");
```

```
                printNodes(list_of_Hajo, writer);
                printNodes(list_of_Kapitany, writer);
                printNodes(list_of_Rakomany, writer);
                printNodes(list_of_Ut, writer);
                printNodes(list_of_Kikoto, writer);
                printNodes(list_of_Megtett_Ut, writer);
                printNodes(list_of_Ut_Kikotoi, writer);

                System.out.println("</" + rootName + ">");
                writer.append("</" + rootName + ">");

                writer.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        private static String indentHandler(int indent) {
            StringBuilder tab = new StringBuilder();
            for (int i = 0; i < indent; i++) {
                tab.append("\t");
            }
            return tab.toString();
        }
}
```

Megjegyzés: pont úgy mint ezelőtt, néhány helyen el van csúszva a kód a mérete miatt.