

Jegyzőkönyv

Adatkezelés XML környezetben

Féléves feladat

Vonatjegy nyilvántartás

Készítette: Molnár Balázs

Neptunkód: KFIXBJ

Dátum: 2022.11.28.

Tartalomjegyzék

A feladat leírása	2
1. feladat.....	3
1. a) feladat – ER modell	3
1. b) feladat – XDM modell.....	3
1. c) feladat – XML dokumentum.....	3
1. d) feladat – XMLSchema.....	6
2. feladat.....	9
2. a) feladat – Adatolvasás	9
2. b) feladat – Adatmódosítás	16
2. c) feladat – Adatlekérdezés	18

A feladat leírása

Utások vonatjegyeit tartjuk számon aszerint, hogy melyik vonaton melyik két állomás között utaznak, és amennyiben helyjegyet váltanak, akkor melyik kocsiban, melyik helyen utaznak.

Vonatnak kocsik csoportját tekintjük, amelyek egy adott vonalon egy adott nap adott időpontjában indulnak. Minden vonatnak van azonosítója. Mivel az időpontot is figyelembe vesszük, ha ugyanazok a kocsik közlekednek egy vonatként két külön időpontban, akkor azt két külön vonatnak tekintjük külön azonosítóval. Így átláthatóbb, hogy melyik utasnak pontosan mikor, melyik vonatra szól a jegye. Az azonosító a VID nevű attribútum, a vonalat a vonalszám jelöli, számontartjuk az indulási és érkezési időpontot is. Ezen kívül egy logikai érték jelzi, hogy a vonat helyjegy köteles-e.

A kocsi egy gyenge egyed, a “része” azonosító kapcsolat köti a vonat egyedhez, a vonat azonosítója és a kocsiszám (partial key) együtt azonosítják a kocsit, illetve rendelkezik egy kocsiosztály attribútummal.

Egy vonat több állomást is érinthet. Ha egy adott állomást érint, akkor az “érinti” kapcsolat köti az adott állomás egyedhez. Ha a vonat egy adott állomásról indul vagy oda érkezik utoljára, akkor a kapcsolat végállomás attribútuma (logikai érték) igaz.

Az állomás egyed egy azonosítóval, egy összetett cím attribútummal és egy névvel rendelkezik.

Ezen kívül bármely két állomás között fennáll egy “távolsága” kapcsolat, melynek távolság attribútuma a két állomás közti távolság km-ben.

Az utas egyed az utasok adatait tárolja, attribútumai egy azonosító, az utas neve és egy vagy több email címe.

A Jegy egyed kapcsolatban van az utassal, a vonattal, illetve két kapcsolat (“állomás1” és “állomás2”) jelzi, hogy melyik két állomás között lehet utazni a jeggyel. A jegynek van egy azonosítója és egy ár attribútuma.

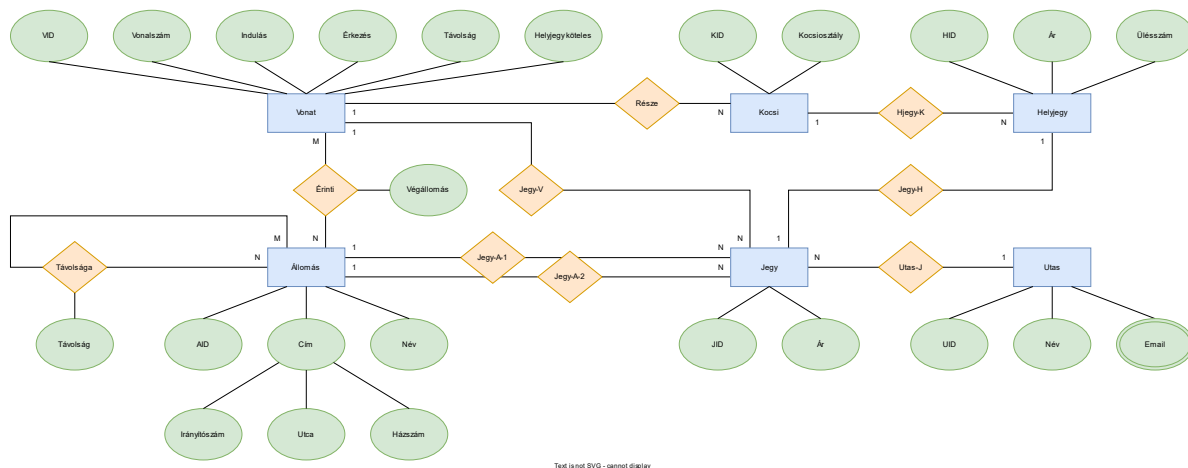
Egyes vonatokon helyjegyet is kell váltani, ezért van a helyjegy egyeddel is kapcsolata a jegy egyednek.

A Helyjegy attribútumai az azonosító és az ár, ezen kívül kapcsolata van azzal a kocsival, ahol az utas utazik. Az ülészsám attribútum jelöli, hogy az utas a kocsiban melyik helyen ül.

1. feladat

1. a) feladat – ER modell

Az adatbázist a feladat leírásának megfelelő ER modellel szemléltetjük.



1. b) feladat – XDM modell

1. c) feladat – XML dokumentum

Az XDM modell alapján elkészítjük az XML dokumentumot.

```
<?xml version="1.0" encoding="UTF-8"?>
<vonatjegyek xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="XMLSchemaKFIXBJ.xsd">
  <!-- Állomások -->
  <allomas aid="a1">
    <iranyitoszam>1087</iranyitoszam>
    <utca>Kerepesi út</utca>
    <hazszam>2-4.</hazszam>
    <nev>Budapest-Keleti</nev>
  </allomas>
  <allomas aid="a2">
    <iranyitoszam>3530</iranyitoszam>
    <utca>Kandó Kálmán tér</utca>
    <hazszam>1-3.</hazszam>
    <nev>Miskolc-Tiszai</nev>
  </allomas>
  <allomas aid="a3">
    <iranyitoszam>6700</iranyitoszam>
    <utca>Tisza Lajos körút</utca>
    <hazszam>28-30.</hazszam>
    <nev>Szeged</nev>
  </allomas>
  <allomas aid="a4">
    <iranyitoszam>7623</iranyitoszam>
    <utca>Indóház tér</utca>
    <hazszam>1.</hazszam>
    <nev>Pécs</nev>
  </allomas>
</vonatjegyek>
```

```
<!-- Állomások távolsága -->
<tavolsaga aid1="a1" aid2="a2">
  <tavolsag>147</tavolsag>
</tavolsaga>
<tavolsaga aid1="a1" aid2="a3">
  <tavolsag>162</tavolsag>
</tavolsaga>
<tavolsaga aid1="a1" aid2="a4">
  <tavolsag>170</tavolsag>
</tavolsaga>

<!-- Vonatok -->
<vonat vid="v1">
  <vonalszam>1</vonalszam>
  <indulas>2022-11-29T08:00:00</indulas>
  <erkezes>2022-11-29T10:30:00</erkezes>
  <tavolsag>200</tavolsag>
  <helyjegy-koteles>>false</helyjegy-koteles>
</vonat>
<vonat vid="v2">
  <vonalszam>2</vonalszam>
  <indulas>2022-11-29T09:00:00</indulas>
  <erkezes>2022-11-29T11:00:00</erkezes>
  <tavolsag>250</tavolsag>
  <helyjegy-koteles>>true</helyjegy-koteles>
</vonat>
<vonat vid="v3">
  <vonalszam>3</vonalszam>
  <indulas>2022-11-29T11:00:00</indulas>
  <erkezes>2022-11-29T14:30:00</erkezes>
  <tavolsag>300</tavolsag>
  <helyjegy-koteles>>true</helyjegy-koteles>
</vonat>

<!-- Kocsik -->
<kocsi vid="v1" kid="k12">
  <kocsiosztaly>1</kocsiosztaly>
</kocsi>
<kocsi vid="v1" kid="k13">
  <kocsiosztaly>2</kocsiosztaly>
</kocsi>
<kocsi vid="v1" kid="k14">
  <kocsiosztaly>2</kocsiosztaly>
</kocsi>
<kocsi vid="v2" kid="k44">
  <kocsiosztaly>1</kocsiosztaly>
</kocsi>
<kocsi vid="v2" kid="k45">
  <kocsiosztaly>1</kocsiosztaly>
</kocsi>
<kocsi vid="v2" kid="k46">
  <kocsiosztaly>2</kocsiosztaly>
</kocsi>
<kocsi vid="v3" kid="k76">
  <kocsiosztaly>1</kocsiosztaly>
</kocsi>
<kocsi vid="v3" kid="k77">
  <kocsiosztaly>1</kocsiosztaly>
</kocsi>
<kocsi vid="v3" kid="k78">
  <kocsiosztaly>2</kocsiosztaly>
```

```
</kocsi>

<!-- Vonatok állomásai -->
<erinti vid="v1" aid="a1">
  <vegallomas>true</vegallomas>
</erinti>
<erinti vid="v1" aid="a2">
  <vegallomas>false</vegallomas>
</erinti>
<erinti vid="v1" aid="a3">
  <vegallomas>true</vegallomas>
</erinti>
<erinti vid="v2" aid="a1">
  <vegallomas>true</vegallomas>
</erinti>
<erinti vid="v2" aid="a2">
  <vegallomas>false</vegallomas>
</erinti>
<erinti vid="v2" aid="a3">
  <vegallomas>false</vegallomas>
</erinti>
<erinti vid="v2" aid="a4">
  <vegallomas>true</vegallomas>
</erinti>

<!-- Utasok -->
<utas uid="u1">
  <nev>Szőke Alex</nev>
  <email>szalex@email.com</email>
  <email>szoke.alex@email.com</email>
</utas>
<utas uid="u2">
  <nev>Farkas Blanka</nev>
  <email>fblanka@email.com</email>
</utas>
<utas uid="u3">
  <nev>Boros Máté</nev>
  <email>bmate@email.com</email>
</utas>
<utas uid="u4">
  <nev>Pap Hanna</nev>
  <email>phanna@email.com</email>
  <email>pap.hanna@email.com</email>
</utas>

<!-- Jegyek -->
<jegy jid="j1">
  <utas>u1</utas>
  <vonat>v1</vonat>
  <allomas1>a1</allomas1>
  <allomas2>a2</allomas2>
  <ar>700</ar>
</jegy>
<jegy jid="j2">
  <utas>u2</utas>
  <vonat>v3</vonat>
  <allomas1>a2</allomas1>
  <allomas2>a4</allomas2>
  <ar>650</ar>
  <helyjegy hid="h1">
    <kocsi>k76</kocsi>
    <ulesszam>12</ulesszam>
```

```

        <ar>150</ar>
    </helyjegy>
</jegy>
<jegy jid="j3">
    <utas>u4</utas>
    <vonat>v2</vonat>
    <allomas1>a3</allomas1>
    <allomas2>a4</allomas2>
    <ar>700</ar>
</jegy>
</vonatjegyek>

```

1. d) feladat – XMLSchema

Az XML fájl alapján XSD fájl készítünk a dokumentum validálásához.

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="vonatjegyek">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="allomas" type="allomasTipus" maxOccurs="unbounded" />
                <xs:element name="tavolsaga" type="tavolsagaTipus" maxOccurs="unbounded"
/>

                <xs:element name="vonat" type="vonatTipus" maxOccurs="unbounded" />
                <xs:element name="kocsi" type="kocsiTipus" maxOccurs="unbounded" />
                <xs:element name="erinti" type="erintiTipus" maxOccurs="unbounded" />
                <xs:element name="utas" type="utasTipus" maxOccurs="unbounded" />
                <xs:element name="jegy" type="jegyTipus" maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>

        <xs:key name="allomas">
            <xs:selector xpath="allomas" />
            <xs:field xpath="@aid" />
        </xs:key>

        <xs:keyref name="tavolsagaAllomas1" refer="allomas">
            <xs:selector xpath="tavolsaga" />
            <xs:field xpath="@aid1" />
        </xs:keyref>
        <xs:keyref name="tavolsagaAllomas2" refer="allomas">
            <xs:selector xpath="tavolsaga" />
            <xs:field xpath="@aid2" />
        </xs:keyref>

        <xs:key name="vonat">
            <xs:selector xpath="vonat" />
            <xs:field xpath="@vid" />
        </xs:key>

        <xs:key name="kocsi">
            <xs:selector xpath="kocsi" />
            <xs:field xpath="@kid" />
        </xs:key>
        <xs:keyref name="kocsiVonat" refer="vonat">
            <xs:selector xpath="kocsi" />
            <xs:field xpath="@vid" />
        </xs:keyref>

        <xs:keyref name="erintiVonat" refer="vonat">

```

```

        <xs:selector xpath="erinti" />
        <xs:field xpath="@vid" />
    </xs:keyref>
    <xs:keyref name="erintiAllomas" refer="allomas">
        <xs:selector xpath="erinti" />
        <xs:field xpath="@aid" />
    </xs:keyref>

    <xs:key name="utas">
        <xs:selector xpath="utas" />
        <xs:field xpath="@uid" />
    </xs:key>

    <xs:keyref name="jegyUtas" refer="utas">
        <xs:selector xpath="jegy" />
        <xs:field xpath="utas" />
    </xs:keyref>
    <xs:keyref name="jegyVonat" refer="vonat">
        <xs:selector xpath="jegy" />
        <xs:field xpath="vonat" />
    </xs:keyref>
    <xs:keyref name="jegyAllomas1" refer="allomas">
        <xs:selector xpath="jegy" />
        <xs:field xpath="allomas1" />
    </xs:keyref>
    <xs:keyref name="jegyAllomas2" refer="allomas">
        <xs:selector xpath="jegy" />
        <xs:field xpath="allomas2" />
    </xs:keyref>

    <xs:keyref name="helyjegyKocsi" refer="kocsi">
        <xs:selector xpath="jegy/helyjegy" />
        <xs:field xpath="kocsi" />
    </xs:keyref>
</xs:element>

<xs:complexType name="allomasTipus">
    <xs:sequence>
        <xs:element name="iranyitoszam" type="xs:integer" />
        <xs:element name="utca" type="xs:string" />
        <xs:element name="hazszam" type="xs:string" />
        <xs:element name="nev" type="xs:string" />
    </xs:sequence>

    <xs:attribute name="aid" type="xs:ID" use="required" />
</xs:complexType>

<xs:complexType name="tavolsagaTipus">
    <xs:sequence>
        <xs:element name="tavolsag" type="xs:integer" />
    </xs:sequence>

    <xs:attribute name="aid1" type="xs:IDREF" use="required" />
    <xs:attribute name="aid2" type="xs:IDREF" use="required" />
</xs:complexType>

<xs:complexType name="vonatTipus">
    <xs:sequence>
        <xs:element name="vonalszam" type="xs:integer" />
        <xs:element name="indulas" type="xs:dateTime" />
        <xs:element name="erkezes" type="xs:dateTime" />
        <xs:element name="tavolsag" type="xs:integer" />
    </xs:sequence>

```

```

        <xs:element name="helyjegy-koteles" type="xs:boolean" />
    </xs:sequence>

    <xs:attribute name="vid" type="xs:ID" use="required" />
</xs:complexType>

<xs:complexType name="kocsiTipus">
    <xs:sequence>
        <xs:element name="kocsiosztaly" type="xs:integer" />
    </xs:sequence>

    <xs:attribute name="vid" type="xs:IDREF" use="required" />
    <xs:attribute name="kid" type="xs:ID" use="required" />
</xs:complexType>

<xs:complexType name="erintiTipus">
    <xs:sequence>
        <xs:element name="vegallomas" type="xs:boolean" />
    </xs:sequence>

    <xs:attribute name="vid" type="xs:IDREF" use="required" />
    <xs:attribute name="aid" type="xs:IDREF" use="required" />
</xs:complexType>

<xs:complexType name="utasTipus">
    <xs:sequence>
        <xs:element name="nev" type="xs:string" />
        <xs:element name="email" type="xs:string" maxOccurs="unbounded" />
    </xs:sequence>

    <xs:attribute name="uid" type="xs:ID" use="required" />
</xs:complexType>

<xs:complexType name="jegyTipus">
    <xs:sequence>
        <xs:element name="utas" type="xs:IDREF" />
        <xs:element name="vonat" type="xs:IDREF" />
        <xs:element name="allomas1" type="xs:IDREF" />
        <xs:element name="allomas2" type="xs:IDREF" />
        <xs:element name="ar" type="xs:integer" />

        <xs:element name="helyjegy" type="helyjegyTipus" minOccurs="0" />
    </xs:sequence>

    <xs:attribute name="jid" type="xs:ID" use="required" />
</xs:complexType>

<xs:complexType name="helyjegyTipus">
    <xs:sequence>
        <xs:element name="kocsi" type="xs:IDREF" />
        <xs:element name="uleszam" type="xs:integer" />
        <xs:element name="ar" type="xs:integer" />
    </xs:sequence>

    <xs:attribute name="hid" type="xs:ID" use="required" />
</xs:complexType>
</xs:schema>

```


2. feladat

2. a) feladat – Adatolvasás

Az adatokat beolvassuk az xml fájlból és struktúráltan kiírjuk a konzolra, illetve az XMLKFIXBJ.txt fájlba.

```
package hu.domparse.kfixbj;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.PrintStream;

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

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

import org.xml.sax.SAXException;

public class DomReadKFIXBJ {
    public static void main(String[] args) throws ParserConfigurationException,
        IOException, SAXException {
        Document doc = parseXMLFile(new File("../XMLKFIXBJ.xml"), new
        File("../XMLSchemaKFIXBJ.xsd"));
        printDocument(doc);

        PrintStream out = new PrintStream(new FileOutputStream("XMLKFIXBJ.txt"));
        System.setOut(out);
        printDocument(doc);
    }

    public static Document parseXMLFile(File xmlFile, File xsdFile)
        throws ParserConfigurationException, IOException, SAXException {
        SchemaFactory schemaFactory =
        SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
        Schema schema = schemaFactory.newSchema(xsdFile);

        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setSchema(schema);
        factory.setNamespaceAware(true);
        factory.setIgnoringElementContentWhitespace(true);
        DocumentBuilder builder = factory.newDocumentBuilder();

        Document doc = builder.parse(xmlFile);
        doc.getDocumentElement().normalize();

        return doc;
    }

    private static void printDocument(Document doc) {
        System.out.println("Vonathely név: " + doc.getElementsByTagName("Vonatjegy").item(0).getTextContent());
    }
}
```

```

System.out.println("=====");
System.out.println();

NodeList nodeList = doc.getDocumentElement().getChildNodes();

for (int i = 0; i < nodeList.getLength(); i++) {
    Node node = nodeList.item(i);

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        Element elem = (Element) node;

        switch (elem.getTagName()) {
            case "allomas" -> printAllomas(elem);
            case "tavolsaga" -> printTavolsaga(elem);
            case "vonat" -> printVonat(elem);
            case "kocsi" -> printKocsi(elem);
            case "erinti" -> printErinti(elem);
            case "utas" -> printUtas(elem);
            case "jegy" -> printJegy(elem);
        }
    }
}

private static void printAllomas(Element elem) {
    System.out.println("Állomás");
    System.out.println("-----");

    String id = elem.getAttribute("aid");

    System.out.printf("ID: %s\n", id);

    String iranyitoszam =
elem.getElementsByTagName("iranyitoszam").item(0).getTextContent();
    String utca = elem.getElementsByTagName("utca").item(0).getTextContent();
    String hazszam = elem.getElementsByTagName("hazszam").item(0).getTextContent();
    String nev = elem.getElementsByTagName("nev").item(0).getTextContent();

    System.out.printf("Irányítószám: %s\n", iranyitoszam);
    System.out.printf("Utca: %s\n", utca);
    System.out.printf("Házszám: %s\n", hazszam);
    System.out.printf("Név: %s\n", nev);

    System.out.println();
}

private static void printTavolsaga(Element elem) {
    String aid1 = elem.getAttribute("aid1");
    String aid2 = elem.getAttribute("aid2");

    String tavolsag =
elem.getElementsByTagName("tavolsag").item(0).getTextContent();

    System.out.printf("%s és %s állomás távolsága: %s\n", aid1, aid2, tavolsag);

    System.out.println();
}

private static void printVonat(Element elem) {
    System.out.println("Vonat");
    System.out.println("-----");
}

```

```

        String id = elem.getAttribute("vid");

        System.out.printf("ID: %s%n", id);

        String vonalszam =
elem.getElementsByTagName("vonalszam").item(0).getTextContent();
        String indulas = elem.getElementsByTagName("indulas").item(0).getTextContent();
        String erkezes = elem.getElementsByTagName("erkezes").item(0).getTextContent();
        String tavolsag =
elem.getElementsByTagName("tavolsag").item(0).getTextContent();
        String helyjegyKoteles = elem.getElementsByTagName("helyjegy-
koteles").item(0).getTextContent();

        System.out.printf("Vonalszám: %s%n", vonalszam);
        System.out.printf("Indulás: %s%n", indulas);
        System.out.printf("Érkezés: %s%n", erkezes);
        System.out.printf("Távolság: %s%n", tavolsag);
        System.out.printf("Helyjegy köteles: %s%n", helyjegyKoteles);

        System.out.println();
    }

    private static void printKocsi(Element elem) {
        System.out.println("Kocsi");
        System.out.println("-----");

        String vid = elem.getAttribute("vid");
        String kid = elem.getAttribute("kid");

        System.out.printf("Vonat ID: %s%n", vid);
        System.out.printf("Kocsi ID: %s%n", kid);

        String kocsiosztaly =
elem.getElementsByTagName("kocsiosztaly").item(0).getTextContent();

        System.out.printf("Kocsiosztály: %s%n", kocsiosztaly);

        System.out.println();
    }

    private static void printErinti(Element elem) {
        String vid = elem.getAttribute("vid");
        String aid = elem.getAttribute("aid");

        String vegallomas =
elem.getElementsByTagName("vegallomas").item(0).getTextContent();

        System.out.printf("%s vonat érinti %s állomást (végállomás: %s)%n", vid, aid,
vegallomas);

        System.out.println();
    }

    private static void printUtas(Element elem) {
        System.out.println("Utas");
        System.out.println("----");

        String id = elem.getAttribute("uid");

        System.out.printf("ID: %s%n", id);

        String nev = elem.getElementsByTagName("nev").item(0).getTextContent();

```

```

        System.out.printf("Név: %s\n", nev);

        NodeList emails = elem.getElementsByTagName("email");

        for (int i = 0; i < emails.getLength(); i++) {
            String email = emails.item(0).getTextContent();
            System.out.printf("Email: %s\n", email);
        }

        System.out.println();
    }

    private static void printJegy(Element elem) {
        System.out.println("Jegy");
        System.out.println("----");

        String id = elem.getAttribute("jid");

        System.out.printf("ID: %s\n", id);

        String utas = elem.getElementsByTagName("utas").item(0).getTextContent();
        String vonat = elem.getElementsByTagName("vonat").item(0).getTextContent();
        String allomas1 =
elem.getElementsByTagName("allomas1").item(0).getTextContent();
        String allomas2 =
elem.getElementsByTagName("allomas2").item(0).getTextContent();
        String ar = elem.getElementsByTagName("ar").item(0).getTextContent();

        System.out.printf("Utas: %s\n", utas);
        System.out.printf("Vonat: %s\n", vonat);
        System.out.printf("Indulási állomás: %s\n", allomas1);
        System.out.printf("Érkezési állomás: %s\n", allomas2);
        System.out.printf("Ár: %s\n", ar);

        Node helyjegy = elem.getElementsByTagName("helyjegy").item(0);

        if (helyjegy != null) {
            Element helyjegyElem = (Element) helyjegy;
            printHelyjegy(helyjegyElem);
        }

        System.out.println();
    }

    private static void printHelyjegy(Element elem) {
        String indent = " ";

        System.out.println("Helyjegy:");

        String id = elem.getAttribute("hid");

        System.out.printf("%sID: %s\n", indent, id);

        String kocsi = elem.getElementsByTagName("kocsi").item(0).getTextContent();
        String ulesszam =
elem.getElementsByTagName("ulesszam").item(0).getTextContent();
        String ar = elem.getElementsByTagName("ar").item(0).getTextContent();

        System.out.printf("%sKocsi: %s\n", indent, kocsi);
        System.out.printf("%sÜlésszám: %s\n", indent, ulesszam);
    }

```

```
        System.out.printf("%sÁr: %s%n", indent, ar);
    }
}
```

A kimenet:

Vonatjegy nyilvántartás
=====

Állomás

ID: a1
Irányítószám: 1087
Utca: Kerepesi út
Házszám: 2-4.
Név: Budapest-Keleti

Állomás

ID: a2
Irányítószám: 3530
Utca: Kandó Kálmán tér
Házszám: 1-3.
Név: Miskolc-Tiszai

Állomás

ID: a3
Irányítószám: 6700
Utca: Tisza Lajos körút
Házszám: 28-30.
Név: Szeged

Állomás

ID: a4
Irányítószám: 7623
Utca: Indóház tér
Házszám: 1.
Név: Pécs

a1 és a2 állomás távolsága: 147

a1 és a3 állomás távolsága: 162

a1 és a4 állomás távolsága: 170

Vonat

ID: v1
Vonalszám: 1
Indulás: 2022-11-29T08:00:00
Érkezés: 2022-11-29T10:30:00
Távolság: 200
Helyjegy köteles: false

Vonat

ID: v2
Vonalszám: 2
Indulás: 2022-11-29T09:00:00
Érkezés: 2022-11-29T11:00:00

Távolság: 250
Helyjegy köteles: true

Vonat

ID: v3

Vonalszám: 3

Indulás: 2022-11-29T11:00:00

Érkezés: 2022-11-29T14:30:00

Távolság: 300

Helyjegy köteles: true

Kocsi

Vonat ID: v1

Kocsi ID: k12

Kocsiosztály: 1

Kocsi

Vonat ID: v1

Kocsi ID: k13

Kocsiosztály: 2

Kocsi

Vonat ID: v1

Kocsi ID: k14

Kocsiosztály: 2

Kocsi

Vonat ID: v2

Kocsi ID: k44

Kocsiosztály: 1

Kocsi

Vonat ID: v2

Kocsi ID: k45

Kocsiosztály: 1

Kocsi

Vonat ID: v2

Kocsi ID: k46

Kocsiosztály: 2

Kocsi

Vonat ID: v3

Kocsi ID: k76

Kocsiosztály: 1

Kocsi

Vonat ID: v3

Kocsi ID: k77

Kocsiosztály: 1

Kocsi

Vonat ID: v3
Kocsi ID: k78
Kocsiosztály: 2

v1 vonat érinti a1 állomást (végállomás: true)
v1 vonat érinti a2 állomást (végállomás: false)
v1 vonat érinti a3 állomást (végállomás: true)
v2 vonat érinti a1 állomást (végállomás: true)
v2 vonat érinti a2 állomást (végállomás: false)
v2 vonat érinti a3 állomást (végállomás: false)
v2 vonat érinti a4 állomást (végállomás: true)

Utas

ID: u1
Név: Szőke Alex
Email: szalex@email.com
Email: szalex@email.com

Utas

ID: u2
Név: Farkas Blanka
Email: fblanka@email.com

Utas

ID: u3
Név: Boros Máté
Email: bmate@email.com

Utas

ID: u4
Név: Pap Hanna
Email: phanna@email.com
Email: phanna@email.com

Jegy

ID: j1
Utas: u1
Vonat: v1
Indulási állomás: a1
Érkezési állomás: a2
Ár: 700

Jegy

ID: j2
Utas: u2
Vonat: v3
Indulási állomás: a2
Érkezési állomás: a4
Ár: 650
Helyjegy:

ID: h1
Kocsi: k76
Ülésszám: 12
Ár: 150

Jegy

ID: j3
Utas: u4
Vonat: v2
Indulási állomás: a3
Érkezési állomás: a4
Ár: 700

2. b) feladat – Adatmódosítás

Módosításokat hajtunk végre az xml dokumentumban.

A módosítások a következők:

1. Az állomások távolságát megkétszerezzük.
2. Megemeljük a jegyek árát.
3. Hozzáadunk egy kocsit az egyik vonathoz.
4. Az egyik vonat útvonalához hozzáadunk egy állomást.
5. Az egyik utasnak megadunk még egy email címet.

Az eredményt kiírjuk az XMLKFIXBJ.mod.xml fájlba és a konzolra is.

```
package hu.domparse.kfixbj;

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

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

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

import static hu.domparse.kfixbj.DomQueryKFIXBJ.*;

public class DomModifyKFIXBJ {
    public static void main(String[] args)
        throws ParserConfigurationException, TransformerException, IOException, SAXException
    {
        Document doc = DomReadKFIXBJ.parseXMLFile(new File("../XMLKFIXBJ.xml"), new
File("../XMLSchemaKFIXBJ.xsd"));

        // Állomások távolságának megkétszerezése
        modify1(doc);

        // Jegyek árának megemelése
        modify2(doc);

        // Kocsi hozzáadása
```



```

        modify3(doc);

        // Állomás hozzárendelése vonathoz
        modify4(doc);

        // Email cím hozzáadása utashoz
        modify5(doc);

        writeXMLFile(doc, new File("../XMLKFIXBJ.mod.xml"));
    }

    private static void modify1(Document doc) {
        NodeList tavolsagaElements = doc.getElementsByTagName("tavolsaga");

        for (int i = 0; i < tavolsagaElements.getLength(); i++) {
            Element tavolsaga = (Element) tavolsagaElements.item(i);

            Element tavolsagElement = firstChild(tavolsaga, "tavolsag");
            int tavolsag = Integer.parseInt(tavolsagElement.getTextContent());
            tavolsagElement.setTextContent(Integer.valueOf(tavolsag * 2).toString());
        }
    }

    private static void modify2(Document doc) {
        NodeList jegyElements = doc.getElementsByTagName("jegy");

        for (int i = 0; i < jegyElements.getLength(); i++) {
            Element jegy = (Element) jegyElements.item(i);

            Element arElement = firstChild(jegy, "ar");
            int ar = Integer.parseInt(arElement.getTextContent());
            arElement.setTextContent(Integer.valueOf(ar + 100).toString());
        }
    }

    private static void modify3(Document doc) {
        Element vonatjegyek = doc.getDocumentElement();

        Element kocsi = doc.createElement("kocsi");
        kocsi.setAttribute("vid", "v3");
        kocsi.setAttribute("kid", "k79");

        Element kocsiosztaly = doc.createElement("kocsiosztaly");
        kocsiosztaly.setTextContent("1");
        kocsi.appendChild(kocsiosztaly);

        vonatjegyek.insertBefore(kocsi, firstChild(vonatjegyek, "kocsi"));
    }

    private static void modify4(Document doc) {
        Element vonatjegyek = doc.getDocumentElement();

        Element erinti = doc.createElement("erinti");
        erinti.setAttribute("vid", "v3");
        erinti.setAttribute("aid", "a4");

        Element vegallomas = doc.createElement("vegallomas");
        vegallomas.setTextContent("true");
        erinti.appendChild(vegallomas);

        vonatjegyek.insertBefore(erinti, firstChild(vonatjegyek, "erinti"));
    }
}

```

```

private static void modify5(Document doc) {
    // Utas azonosítója
    String uid = "u3";
    Element utas = getElementById(doc, "utas", "uid", uid);

    // Utas új email címe
    String newEmail = "boros.mate@email.com";
    Element email = doc.createElement("email");
    email.setTextContent(newEmail);
    utas.insertBefore(email, firstChild(utas, "email"));
}

private static void writeXMLFile(Document doc, File outputFile) throws
TransformerException {
    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();

    transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");

    DOMSource domSource = new DOMSource(doc);

    StreamResult console = new StreamResult(System.out);
    StreamResult file = new StreamResult(outputFile);

    transformer.transform(domSource, console);
    transformer.transform(domSource, file);
}
}

```

2. c) feladat – Adatlekérdezés

Lekérdezéseket hajtunk végre az xml fájlban.

A lekérdezések a következők:

1. Jegy ID-ja alapján lekérdezzük egy utazást.
2. Lekérdezzük azokat a vonatokat, amelyek érintenek egy adott állomást.
3. Lekérdezzük, hogy adott állomáson melyik napokon halad el vonat.
4. Lekérdezzük egy vonat menetidejét.
5. Lekérdezzük azokat az állomásokat, amelyek 165 km-nél közelebb vannak.

Az eredményt kiírjuk a konzolra.

```

package hu.domparse.kfixbj;

import java.io.File;
import java.io.IOException;
import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.ParserConfigurationException;

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

```

```

import org.xml.sax.SAXException;

public class DomQueryKFIXBJ {
    public static void main(String[] args) throws ParserConfigurationException,
    IOException, SAXException {
        Document doc = DomReadKFIXBJ.parseXMLFile(new File("../XMLKFIXBJ.xml"), new
        File("../XMLSchemaKFIXBJ.xsd"));

        String jid;
        String aid;
        String vid;

        // Utazás lekérdezése jegy ID-ja alapján
        jid = "j1";
        query1(doc, jid);

        // Mely vonat érint adott állomást
        aid = "a3";
        query2(doc, aid);

        // Adott állomást mely napokon érint vonat
        aid = "a2";
        query3(doc, aid);

        // Adott vonat menetideje
        vid = "v3";
        query4(doc, vid);

        // Állomások, amelyek 165 km-nél közelebb vannak
        query5(doc);
    }

    private static void query1(Document doc, String jid) {
        String indent = " ";

        Element jegy = getElementById(doc, "jegy", "jid", jid);

        String uid = firstChildTextContent(jegy, "utas");
        Element utas = getElementById(doc, "utas", "uid", uid);
        String utasNev = firstChildTextContent(utas, "nev");

        String aid1 = firstChildTextContent(jegy, "allomas1");
        String aid2 = firstChildTextContent(jegy, "allomas2");
        Element allomas1 = getElementById(doc, "allomas", "aid", aid1);
        Element allomas2 = getElementById(doc, "allomas", "aid", aid2);
        String allomasNev1 = firstChildTextContent(allomas1, "nev");
        String allomasNev2 = firstChildTextContent(allomas2, "nev");

        System.out.printf("%s (ID: %s) utazása:%n", utasNev, uid);
        System.out.printf("%sIndulási állomás: %s%n", indent, allomasNev1);
        System.out.printf("%sÉrkezési állomás: %s%n", indent, allomasNev2);

        System.out.println();
    }

    private static void query2(Document doc, String aid) {
        Element allomas = getElementById(doc, "allomas", "aid", aid);
        String allomasNev = firstChildTextContent(allomas, "nev");

        NodeList erintiElements = doc.getElementsByTagName("erinti");
    }

```

```

        for (int i = 0; i < erintiElements.getLength(); i++) {
            Element erinti = (Element) erintiElements.item(i);

            if (erinti.getAttribute("aid").equals(aid)) {
                String vid = erinti.getAttribute("vid");
                boolean vegallomas = Boolean.parseBoolean(firstChildTextContent(erinti,
"vegallomas"));

                System.out.printf(
                    "%s vonat érinti %s állomást %s%n",
                    vid, allomasNev,
                    vegallomas ? "(végállomás)" : ""
                );
            }
        }

        System.out.println();
    }

    private static void query3(Document doc, String aid) {
        Element allomas = getElementById(doc, "allomas", "aid", aid);
        String allomasNev = firstChildTextContent(allomas, "nev");

        List<LocalDate> dates = new ArrayList<>();
        NodeList erintiElements = doc.getElementsByTagName("erinti");

        for (int i = 0; i < erintiElements.getLength(); i++) {
            Element erinti = (Element) erintiElements.item(i);

            if (erinti.getAttribute("aid").equals(aid)) {
                String vid = erinti.getAttribute("vid");
                Element vonat = getElementById(doc, "vonat", "vid", vid);
                LocalDateTime dateTime =
LocalDateTime.parse(firstChildTextContent(vonat, "indulas"));
                LocalDate date = dateTime.toLocalDate();

                if (!dates.contains(date)) {
                    dates.add(date);
                    System.out.printf(
                        "%s állomást érinti vonat %tY.%tm.%td. napon%n",
                        allomasNev, date, date, date
                    );
                }
            }
        }

        System.out.println();
    }

    private static void query4(Document doc, String vid) {
        Element vonat = getElementById(doc, "vonat", "vid", vid);

        LocalDateTime indulas = LocalDateTime.parse(firstChildTextContent(vonat,
"indulas"));
        LocalDateTime erkezes = LocalDateTime.parse(firstChildTextContent(vonat,
"erkezes"));
        Duration menetido = Duration.between(indulas, erkezes);

        System.out.printf(
            "%s vonat menetideje: %02d:%02d:%02d%n",
            vid, menetido.toHoursPart(), menetido.toMinutesPart(),
            menetido.toSecondsPart()
        );
    }

```

```

    );

    System.out.println();
}

private static void query5(Document doc) {
    NodeList tavolsagaElements = doc.getElementsByTagName("tavolsaga");

    for (int i = 0; i < tavolsagaElements.getLength(); i++) {
        Element tavolsaga = (Element) tavolsagaElements.item(i);
        int tavolsag = Integer.parseInt(firstChildTextContent(tavolsaga,
"tavolsag"));

        if (tavolsag < 165) {
            String aid1 = tavolsaga.getAttribute("aid1");
            String aid2 = tavolsaga.getAttribute("aid2");
            Element allomas1 = getElementById(doc, "allomas", "aid", aid1);
            Element allomas2 = getElementById(doc, "allomas", "aid", aid2);
            String allomasNev1 = firstChildTextContent(allomas1, "nev");
            String allomasNev2 = firstChildTextContent(allomas2, "nev");

            System.out.printf("%s - %s: %d km%n", allomasNev1, allomasNev2,
tavolsag);
        }
    }

    System.out.println();
}

public static Element getElementById(Document doc, String tagName, String attribute,
String id) {
    NodeList elements = doc.getElementsByTagName(tagName);

    for (int i = 0; i < elements.getLength(); i++) {
        Element elem = (Element) elements.item(i);

        if (elem.getAttribute(attribute).equals(id)) {
            return elem;
        }
    }

    return null;
}

public static Element firstChild(Element elem, String tagName) {
    return (Element) elem.getElementsByTagName(tagName).item(0);
}

public static String firstChildTextContent(Element elem, String tagName) {
    return firstChild(elem, tagName).getTextContent();
}
}

```

A kimenet:

Szóke Alex (ID: u1) utazása:

Indulási állomás: Budapest-Keleti

Érkezési állomás: Miskolc-Tiszai

v1 vonat érinti Szeged állomást (végállomás)

v2 vonat érinti Szeged állomást

Miskolc-Tiszai állomást érinti vonat [2022.11.29.](#) napon

v3 vonat menetideje: [03:30:00](#)

Budapest-Keleti - Miskolc-Tiszai: [147](#) km

Budapest-Keleti - Szeged: [162](#) km