JEGYZŐKÖNYV

Adatkezelés XML környezetben   
Féléves feladat

Autókölcsönző

Készítette: **Pázmán András**

Neptunkód: **H2Z4X3**   
Dátum: 2023.11.29

**Tartalomjegyzék**

[A feladat leírása 3](#_A_feladat_leírása:)

[1. feladat 4](#_1._feladat)

[1a) Az adatbázis ER modell tervezése 4](#_1a)_Az_adatbázis)

[1b) Az adatbázis konvertálása XDM modellre 5](#_1b)_Az_adatbázis)

[1c) Az XDM modell alapján XML dokumentum készítése: 6](#_1c)_Az_XDM)

[1d) Az XML dokumentum alapján XMLSchema készítése 10](#_Toc152143268)

[2. feladat 14](#_Toc152143269)

[2a) adatolvasás 14](#_Toc152143270)

[2b) adatmódosítás 19](#_Toc152143271)

[2c) adatlekérdezés 23](#_Toc152143272)

[2d) adatírás 23](#_Toc152143273)

## A feladat leírása:

A feladat egy autókölcsönző vállalkozáshoz kapcsolódik, amely több telephellyel rendelkezik országszerte.   
A telephelyeken különböző autók találhatóak amelyeket az ügyfelek kibérelhetnek.

A bérlés adminisztrálását és a telep működését az alkalmazottak biztosítják akik különböző pozíciókban dolgoznak . A telephely tulajdonosához tartozik az adott telep melynek felügyeli a működését.  
Egy tulajdonoshoz egy telep tartozik, ő kezeli a telep bevételét. Ezt jelzi a bevétel tulajdonság.   
Minden telepnek más a kapacitása mivel az autókat tárolni kell ezért a fennálló kapacitás korlátozott.  
Tovább a telep rendelkezik egy minősítési számmal (beosztás) melynek értéke A, B, C, D, E szerint oszlik meg.   
Az A kategória a legjobb értékelés, az E a legrosszabb. Az értékelést sok minden befolyásolhatja, akár a vevői értékelés, telep kapacitás stb.

Ha egy személy egy autót szeretne kibérelne akkor az adatait az adatbázisban rögzítjük. Ennek több oka is van, az adatok alapján könnyebb visszakeresni az adott személy ha esetlegesen kár keletkezik az autóba.   
Egy alkalmazott egy időben csak egy autót bérelhet ki, ezáltal egy adott autó is csak 1 ügyfélnél lehet ugyan abban az időpontban. Ezzel kizárjuk azt hogy többen is ugyan azt az autót foglalják re ugyan arra az időpontra.

Az alkalmazottak pozíciójuknak megfelelően szerezhetnek plusz képesítéséket melyekkel növelhetik a fizetésüket. A különböző juttatások 10 000- 20 000- 30 000 Ft összegekben oszlanak meg. A képesítéshez különböző szint és nyelv is tartozik melyek tovább befolyásolják a juttatás mértékét. Egy alkalmazott több képesítéssel is rendelkezhet. A képesítések közé beletartozik a nyelvvizsga bizonyítvány is is.

## 1. feladat

## 1a) Az adatbázis ER modell tervezése

Az alábbi ER modell tartalmazza az autókölcsönző adatmodell felépítését.

A ER modellben 6 egyed található melyek között megtalálható 1:1, 1:N, illetve N:M kapcsolat is.

Az egyedek minimum 4 tulajdonsággal rendelkeznek, ezeken kívül egyedi azonosításul kaptak egy Id-t .

**Tulajdonos:**

* Név (Tulajdonos neve )
* Adószám (Tulajdonos adószáma )
* Bevétel ( A telep bevétele )
* TelSzám ( Tualjdonos telefonszáma )
* TulID ( primary key)

**Telep**:

* Kapacitás ( a tárolható autók száma )
* Cím (összetett tulajdonság, tartalmazza a települsét, utca, házszámot )
* Beosztás ( A telep minősítését, kategoriajat tartalmazza ( A, B, C, D vagy E )
* TelepID ( primary key)

**Autok**:

* Modell (az autó típusát tartalmazza )
* Rendszám ( a bérelhető autó rendszáma)
* Alvázszám ( az autó alvázszáma )
* Gyártási év ( az autó gyártási éve )
* AutoID ( primary key)

**Bérlő**:

* Név ( A bérlő neve )
* TelSzám (A bérlő telefon száma )
* Email (A bérlő email címe )
* Jogosítvány szám ( Tualjdonos telefonszáma )
* BérlőlID ( primary key)

**Alkalmazottak**:

* + Név ( alkalmazott neve )
  + Beosztás ( a feladatkör amit az alkalmazott ellát)
  + SzülÉv (Az alkalmazott születési éve )
  + Életkor ( Az alkalmazott életkora )
  + AlkID ( primary key)

**Képesítés**:

* Típus ( a bizonyítvány típusa )
* Nyelv
* Év
* BizSzám ( bizonyítvány száma )
* KépesítésID ( primary key)

**Bonusz**:

* Extra ( a juttatás mértéke)
* BonuszID ( primary key )

A képen rajz, diagram, vázlat, origami látható

Automatikusan generált leírás

## 1b) Az adatbázis konvertálása XDM modellre

A képen fehér, sor, diagram látható

Automatikusan generált leírásAz XDM modell elkészítésé során kialakításra kerültek a kapcsolatok az elsődleges és idegen kulcsok között. Továbbá a modell kivitelezése alatt ügyleni kellett hogy a kapcsolatokat jelölő szaggatott vonalak ne keresztezzék egymást mivel ez rontja a modell átláthatóságát.

## **1c) Az XDM modell alapján XML dokumentum készítése:**

A képen szöveg, képernyőkép, menü, dokumentum látható

Automatikusan generált leírásAz XDM modell átkonvertálva XML modellre az elsődleges és másodlagos kulcsokból attribútumok lettek. Minden elem legalább 3 példányban kerül átkonvertálásra.

A képen szöveg, képernyőkép, Betűtípus, szám látható

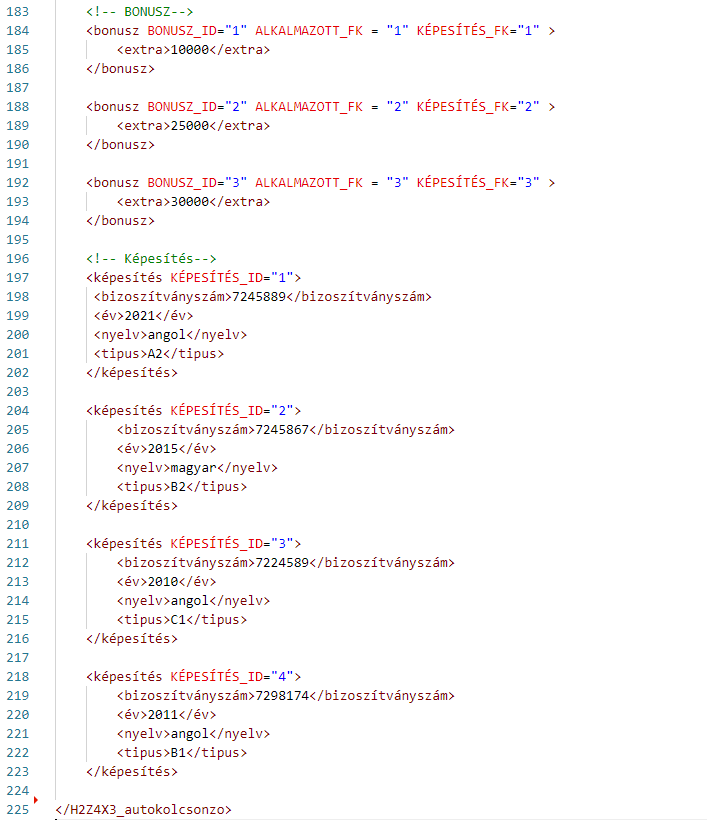
Automatikusan generált leírásA képen szöveg, képernyőkép látható

Automatikusan generált leírás

A képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás



**1d) Az XML dokumentum alapján XMLSchema készítése**  
 **-** saját típusok, ref, key, keyref, speciális elemek.

A képen szöveg, képernyőkép, dokumentum, menü látható

Automatikusan generált leírásXML dokumentumhoz elkészítettem az XML Schemat. Elsődlegesen az egyszerű és   
 saját típusokat gyűjtöttem ki, ezután komplex típusok, majd elsődleges kulcsok, idegen kulcsok.

A képen szöveg, képernyőkép, menü, Betűtípus látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus, szám látható

Automatikusan generált leírás

A képen szöveg, Betűtípus, képernyőkép látható

Automatikusan generált leírásA képen szöveg, képernyőkép, dokumentum, Betűtípus látható

Automatikusan generált leírás

A képen szöveg, képernyőkép látható

Automatikusan generált leírásA képen szöveg, képernyőkép, Betűtípus látható

Automatikusan generált leírás

2. feladat

A feladat egy DOM program készítése az XML dokumentum **- *XMLNeptunkod.xml*** – adatainak adminisztrálása alapján

2a) adatolvasás

Az alábbi Java-kód képes az XML-adatok olvasására és feldolgozására (DOM) használatával.

package hu.domparse.h2z4x3;

import java.io.File;

import java.io.IOException;

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 org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.xml.parsers.\*;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.\*;

import org.xml.sax.SAXException;

public class DomReadH2Z4X3 {

     public static void main(String[] args) {

        //xml read meghívása

        readXMLDocument("XMLh2z4X3.xml");

    }

     /\*

      \* Ez a metodus beolvassa

      \*/

     private static void readXMLDocument(String filePath) {

        try {

            File newXMLFile = new File(filePath);

            StreamResult newXmlStream = new StreamResult(newXMLFile);

            Document document = parseXML("./XMLh2z4X3.xml");

            writeDocument(document, newXmlStream);

            System.out.println(formatXML(document)); //strukturáltan kiiratás

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

    /\*

     \* Az alábbi metodus beolvassa az XML dokumentumot

     \*/

    public static Document parseXML(String fileName) throws  ParserConfigurationException, SAXException, IOException{

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

            DocumentBuilder builder = factory.newDocumentBuilder();

            // XML dokumentum beolvasása és DOM dokumentummá alakítása

            Document document = builder.parse(new File(fileName));

            // metodus meghivas üres szövegek eltávolítására

            cleanDocument(document.getDocumentElement());

            return document;

    }

    /\*

     \*Delete empty lines

     \*

     \*/

    private static void cleanDocument(Node root) {

        NodeList nodeList = root.getChildNodes();

        // Delete empty text spaces

        List<Node> toDeleteEmptyTextList = new ArrayList<>();

        //veigjarni a gyoker elemeket

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

            //megnezi hogy a textnode ures-e

            if (nodeList.item(i).getNodeType() == Node.TEXT\_NODE && nodeList.item(i).getTextContent().strip().isEmpty()) {

                toDeleteEmptyTextList.add(nodeList.item(i)); //hozzadja

            } else {

                cleanDocument(nodeList.item(i));

            }

        }

        for (Node node : toDeleteEmptyTextList) {

            root.removeChild(node);

        }

    }

    /\*

     \* Az alábbi metodus kiirja file-ba az xml dokumentumot

     \*/

    public static void writeDocument(Document document, StreamResult output){

        try {

            TransformerFactory transformerFactory = TransformerFactory.newInstance();

            Transformer transformer = transformerFactory.newTransformer();

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

            transformer.setOutputProperty(OutputKeys.INDENT, "yes");

            // DOMletrehozas dokumentumbol

            DOMSource source = new DOMSource(document);

            transformer.transform(source, output);

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

    /\*

     \* XML formazo metodus

     \*/

    public static String formatXML(Document document) {

        return "<?xml version=\"" + document.getXmlVersion() + "\" encoding=\"" + document.getXmlEncoding() + "\" ?>\n" +

               formatElement(document.getDocumentElement(), 0);

    }

    /\*

     \* Az xml kiirast formazza

     \*/

    public static String formatElement(Node node, int indent) {

        if (node.getNodeType() != Node.ELEMENT\_NODE) {

            return "";

        }

        StringBuilder output = new StringBuilder();

        output.append(getIndent(indent)).append("<").append(((Element) node).getTagName());

        if (node.hasAttributes()) {

            for (int i = 0; i < node.getAttributes().getLength(); i++) {

                Node attribute = node.getAttributes().item(i);

                output.append(" ").append(attribute.getNodeName()).append("=\"").append(attribute.getNodeValue()).append("\"");

            }

        }

        //gyerekek lekerdezese

        NodeList children = node.getChildNodes();

        if (children.getLength() == 1 && children.item(0).getNodeType() == Node.TEXT\_NODE) {

            output.append(">").append(children.item(0).getTextContent().trim()).append("</").append(((Element) node).getTagName()).append(">\n");

        } else {

            output.append(">\n"); //zarotag

            for (int i = 0; i < children.getLength(); i++) {

                output.append(formatElement(children.item(i), indent + 1));

            }

            // zarotag hozzadassa

            output.append(getIndent(indent)).append("</").append(((Element) node).getTagName()).append(">\n");

        }

        return output.toString();

    }

    private static String getIndent(int indent) {

        StringBuilder indentation = new StringBuilder();

        for (int i = 0; i < indent; i++) {

            indentation.append("    ");

        }

        return indentation.toString();

    }

}

2b) adatmódosítás (kód – comment együtt) – fájlnév: ***DOMModifyNeptunkod.java***

Az alábbi Java kód egy DOM alapú XML fájl módosítást valósít meg. A DOM segítségével egy XML fájl tartalmát egy fastruktúrában jeleníti meg , és lehetőséget ad a fájl elemeinek navigálására, módosítására.

package hu.domparse.h2z4x3;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;

import org.w3c.dom.NamedNodeMap;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

public class DomModifyH2Z4X3 {

    public static void main(String[] args) {

        try {

            //xxml file megnyitása adatok boelvasásához

            File file = new File("XMLh2z4x3.xml");

            //dokumentum létrehozasa a beolvasott filebol

            DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();

            Document document = documentBuilder.parse(file);

            //dokumentum normalizásálsa

            document.getDocumentElement().normalize();

            //1. képsítés nyelv módosítása maygarról németre

            NodeList kepesitesNyelvList = document.getElementsByTagName("képesítés");

            for (int temp = 0; temp < kepesitesNyelvList.getLength(); temp++) {

                Node node = kepesitesNyelvList.item(temp);

                if (node.getNodeType() == Node.ELEMENT\_NODE) {

                    NodeList childNodes = node.getChildNodes();

                    for (int j = 0; j < childNodes.getLength(); j++) {

                        Node childNode = childNodes.item(j);

                        if ("nyelv".equals(childNode.getNodeName())) {

                            if ("magyar".equals(childNode.getTextContent())) {

                                childNode.setTextContent("német");

                            }

                        }

                    }

                }

            }

            //2. Bevétel növelése 130 000ft-al Cset Elek-nek.

            NodeList telepBevetelList = document.getElementsByTagName("tulajdonos");

            for (int temp = 0; temp < telepBevetelList.getLength(); temp++) {

                Node node = telepBevetelList.item(temp);

                if (node.getNodeType() == Node.ELEMENT\_NODE) {

                    NodeList childNodes = node.getChildNodes();

                    for (int j = 0; j < childNodes.getLength(); j++) {

                        Node childNode = childNodes.item(j);

                        if ("név".equals(childNode.getNodeName())) {

                            if ("Cset Elek".equals(childNode.getTextContent())) {

                                for (int k = 0; k < childNodes.getLength(); k++) {

                                    childNode = childNodes.item(k);

                                    if ("bevétel".equals(childNode.getNodeName())) {

                                        childNode.setTextContent("1930000");

                                    }

                                }

                            }

                        }

                    }

                }

            }

            //3. Ahol a telep kapacitas 5 módosítsuk 6ra

            NodeList telepKapacatisList = document.getElementsByTagName("telep");

            for (int temp = 0; temp < telepKapacatisList.getLength(); temp++) {

                Node node = telepKapacatisList.item(temp);

                if (node.getNodeType() == Node.ELEMENT\_NODE) {

                    NodeList childNodes = node.getChildNodes();

                    for (int j = 0; j < childNodes.getLength(); j++) {

                        Node childNode = childNodes.item(j);

                        if ("kapacitás".equals(childNode.getNodeName())) {

                            if ("5".equals(childNode.getTextContent())) {

                                childNode.setTextContent("6");

                            }

                        }

                    }

                }

            }

            //4. Az 5. alkalmazott munkahelyének (telepid) módosítása

            NodeList alkalmazottNList = document.getElementsByTagName("alkalmazott");

            for (int temp = 0; temp < alkalmazottNList.getLength(); temp++) {

                Node node = alkalmazottNList.item(temp);

                NamedNodeMap attribute = node.getAttributes();

                Node nodeAttributeID = attribute.getNamedItem("ALK\_ID");

                Node nodeAttributeFK = attribute.getNamedItem("ALK\_TELEP");

                if (nodeAttributeID.getTextContent().equals("5")) {

                    nodeAttributeFK.setTextContent("4");

                }

            }

            //5. Ahol a bónusz kevesebb mint 20000 , az értéket módosítsuk 15000-re

            NodeList bonuszList = document.getElementsByTagName("bonusz");

            for (int temp = 0; temp < bonuszList.getLength(); temp++) {

                Node node = bonuszList.item(temp);

                if (node.getNodeType() == Node.ELEMENT\_NODE) {

                    NodeList childNodes = node.getChildNodes();

                    for (int j = 0; j < childNodes.getLength(); j++) {

                        Node childNode = childNodes.item(j);

                        if ("extra".equals(childNode.getNodeName())) {

                            if (Integer.valueOf(childNode.getTextContent()) <= 20000) {

                                childNode.setTextContent("15000");

                            }

                        }

                    }

                }

            }

            //a modositott adatok kiirasa konzolra

            TransformerFactory transformerFactory = TransformerFactory.newInstance();

            Transformer transformer = transformerFactory.newTransformer();

            DOMSource source = new DOMSource(document);

            System.out.println("---- Modositott fajl -----");

            StreamResult consoleResult = new StreamResult(System.out);

            transformer.transform(source, consoleResult);

        } catch (Exception e ) {

            e.printStackTrace();

        }

    }

}

2c) adatlekérdezés

Az alábbi java kód xml fájlból beolvasott adatok meghatározott lekérdezésére szolgál.  
A lekérdezések gyakran egyéb feltételekhez kötöttek, ezek az attribútumok vagy tulajdonságok értékeihez vannak kötve.   
A lekérdezések eredményeit futás után kiíratjuk a konzolra.

A következő 5 lekérdezés van megvalósítva a programban:

1. Az A minősítésű telepek kapacitásának lekérdezése
2. Az 1500000ft-nál nagyobb bevételű telep tulajdonosának kiirása
3. A '2'-es ID-jű telep autóinak kiiratása amelyek 2020 vagy attól kisebb évjáratúak.
4. Írjuk ki a 'Git Áron' elérhetőségeit
5. A 2-es telep alkalamzottianak az adatai

package hu.domparse.h2z4x3;

import java.io.File;

import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NamedNodeMap;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

public class DomQueryH2Z4X3 {

    public static void main(String[] args) throws IOException, ParserConfigurationException, SAXException {

        try {

            //xml file megnyitasa adatok beolvasasahoz

            File file = new File("XMLh2z4x3.xml");

            //dokumentum letrehozasa  a beolvasott file-ból

            DocumentBuilder documentBuilder = DocumentBuilderFactory.newInstance().newDocumentBuilder();

            Document document = documentBuilder.parse(file);

            //dokumentum normalizálása

            document.getDocumentElement().normalize();

            lekerdezes1(document);

            lekerdezes2(document);

            lekerdezes3(document);

            lekerdezes5(document);

            lekerdezes6(document);

            } catch (IOException e) {

                e.printStackTrace();

            } catch (ParserConfigurationException e) {

                e.printStackTrace();

            } catch (SAXException e) {

                e.printStackTrace();

            }

    }

    public static void lekerdezes1(Document document) {

        try {

            System.out.println("\n1. Az A minősítésű telepek kapacitásának lekérdezése");

             NodeList nList1 = document.getElementsByTagName("telep");

             for (int i = 0; i < nList1.getLength(); i++) {

                Node nNode = nList1.item(i);

                if (nNode.getNodeType()==Node.ELEMENT\_NODE) {

                    Element elem = (Element) nNode;

                    NodeList childNodes = nNode.getChildNodes();

                    for (int j = 0; j < childNodes.getLength(); j++) {

                        Node childNode = childNodes.item(j);

                        if ("minősítés".equals(childNode.getNodeName())) {

                            if ("A".equals(childNode.getTextContent())) {

                                System.out.println("");

                                String telepid = elem.getAttribute("TELEP\_ID");

                                Node node1 = elem.getElementsByTagName("kapacitás").item(0);

                                String node1Text = node1.getTextContent();

                                System.out.println("<telep TELEP\_ID = "  +telepid + " >");

                                System.out.println(" <kapacitás> " + node1Text + " </kapacitás>");

                                System.out.println("</telep>");

                            }

                        }

                    }

                }

             }

             System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

        } catch (NullPointerException e) {

            e.printStackTrace();

        }

    }

    public static void lekerdezes2(Document document) {

        try {

             System.out.println("\n2. Az 1500000ft-nál nagyobb bevételű telep tulajdonosának kiirása");

                NodeList nList2 = document.getElementsByTagName("tulajdonos");

                for (int i = 0; i < nList2.getLength(); i++) {

                    Node nNode = nList2.item(i);

                    if (nNode.getNodeType()==Node.ELEMENT\_NODE) {

                        Element elem = (Element) nNode;

                        NodeList childNodes = nNode.getChildNodes();

                        NamedNodeMap attr = nNode.getAttributes();

                        if (nNode.getNodeType()==Node.ELEMENT\_NODE) {

                            for (int j = 0; j < childNodes.getLength(); j++) {

                                Node childNode = childNodes.item(j);

                                if ("bevétel".equals(childNode.getNodeName())) {

                                    if (Integer.valueOf(childNode.getTextContent()) > 1500000 ) {

                                        System.out.println("");

                                        String tulajdonosid = elem.getAttribute("TUL\_ID");

                                        Node node1 = elem.getElementsByTagName("név").item(0);

                                        String node1Text = node1.getTextContent();

                                        System.out.println("<tulajdonos Tulajdonos\_ID = > " + tulajdonosid + ">");

                                        System.out.println("  <név> " + node1Text + "</név>");

                                        System.out.println("</tulajdonos>");

                                    }

                                }

                            }

                        }

                    }

                }

               System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

        } catch (NullPointerException e) {

            e.printStackTrace();

        }

    }

    public static void lekerdezes3(Document document) {

        try {

              System.out.println("\n3.: A '2'-es ID-jű telep autóinak kiiratása amelyek 2020 vagy attól kisebb évjáratúak.");

                NodeList nList3 = document.getElementsByTagName("autok");

                for (int i = 0; i < nList3.getLength(); i++) {

                    Node nNode = nList3.item(i);

                    if (nNode.getNodeType()==Node.ELEMENT\_NODE) {

                        Element elem = (Element) nNode;

                        NodeList childNodes = nNode.getChildNodes();

                        NamedNodeMap attr = nNode.getAttributes();

                        Node nodeAtribute = attr.getNamedItem("AUTO\_TELEP");

                        if (nodeAtribute.getTextContent().equals("2")) {

                            for (int j = 0; j < childNodes.getLength(); j++) {

                                Node childNode = childNodes.item(j);

                                if ("gyártásiév".equals(childNode.getNodeName())) {

                                    if (Integer.valueOf(childNode.getTextContent()) <= 2020 ) {

                                        System.out.println("");

                                        String autoid = elem.getAttribute("AUTO\_ID");

                                        Node node1 = elem.getElementsByTagName("rendszám").item(0);

                                        String node1Text = node1.getTextContent();

                                        Node node2 = elem.getElementsByTagName("modell").item(0);

                                        String node2Text = node2.getTextContent();

                                        System.out.println("<autok AUTO\_ID =  " + autoid + ">");

                                        System.out.println("  <rensszám> " + node1Text + " <rensszám>");

                                        System.out.println("  <modell> " + node2Text + " </modell>" );

                                        System.out.println("</autok>");

                                    }

                                }

                            }

                        }

                    }

                }

                System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

         } catch (NullPointerException e) {

            e.printStackTrace();

        }

    }

    public static void lekerdezes5(Document document) {

        try {

             System.out.println("\n5. Írjuk ki a 'Git Áron' elérhetőségeit");

                NodeList nList5 = document.getElementsByTagName("bérlő");

                for (int i = 0; i < nList5.getLength(); i++) {

                    Node nNode = nList5.item(i);

                    if (nNode.getNodeType()==Node.ELEMENT\_NODE) {

                        Element elem = (Element) nNode;

                        NodeList childNodes = nNode.getChildNodes();

                        for (int j = 0; j < childNodes.getLength(); j++) {

                            Node childNode = childNodes.item(j);

                            if ("név".equals(childNode.getNodeName())) {

                                    if ("Git Áron".equals(childNode.getTextContent())) {

                                        System.out.println("");

                                        String telepid = elem.getAttribute("BÉRLŐ\_ID");

                                        Node node1 = elem.getElementsByTagName("telszám").item(0);

                                        String node1Text = node1.getTextContent();

                                        Node node2 = elem.getElementsByTagName("email").item(0);

                                        String node2Text = node2.getTextContent();

                                        System.out.println("<bérlő BÉRLŐ\_ID =  " + telepid + " >");

                                        System.out.println("  <telszám> " + node1Text +  " </telszám>");

                                        System.out.println("  <email> " + node2Text + "</email>");

                                        System.out.println("</bérlő>");

                                    }

                            }

                        }

                    }

                }

                System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

         } catch (NullPointerException e) {

            e.printStackTrace();

        }

    }

    public static void lekerdezes6(Document document) {

        try {

              System.out.println("\n6. A 2-es telep alkalamzottianak az adatai");

                NodeList nList6 = document.getElementsByTagName("alkalmazott");

                for (int i = 0; i < nList6.getLength(); i++) {

                    Node nNode = nList6.item(i);

                    if (nNode.getNodeType()==Node.ELEMENT\_NODE) {

                        Element elem = (Element) nNode;

                        NamedNodeMap attr = nNode.getAttributes();

                        Node nodeAttrM = attr.getNamedItem("ALK\_TELEP");

                        if (nodeAttrM.getTextContent().equals("2")) {

                                System.out.println("");

                                String alkalmazottid = elem.getAttribute("ALK\_ID");

                                Node node1 = elem.getElementsByTagName("név").item(0);

                                String node1Text = node1.getTextContent();

                                Node node2 = elem.getElementsByTagName("beosztás").item(0);

                                String node2Text = node2.getTextContent();

                                Node node3 = elem.getElementsByTagName("életkor").item(0);

                                String node3Text = node3.getTextContent();

                                Node node4 = elem.getElementsByTagName("születésiév").item(0);

                                String node4Text = node4.getTextContent();

                                System.out.println("<alkalmazott ALKALMAZOTT\_ID =  " + alkalmazottid + ">");

                                System.out.println("  <név> " + node1Text + " </név>");

                                System.out.println("  <beosztás> " + node2Text + " </beosztás>");

                                System.out.println("  <életkor> " + node3Text +  " </életkor>");

                                System.out.println("  <születésiév> " + node4Text + " </születésiév>");

                                System.out.println("</alkalmazott>");

                            }

                    }

                }

                System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

         } catch (NullPointerException e) {

            e.printStackTrace();

        }

    }

2d) adatírás

Az alábbi java kód létrohozza az új XML file tartalmát, majd egy fastruktúrában reprezentálja és kiírja egy új XML fájlba.

package hu.domparse.h2z4x3;

import java.io.File;

import java.io.FileOutputStream;

import java.io.OutputStream;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerConfigurationException;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

public class DomWriteH2Z4X3 {

    public static void main(String args[]) {

        try {

            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

            DocumentBuilder dbBuilder = dbFactory.newDocumentBuilder();

            Document document = dbBuilder.newDocument();

            // root elements megadasa

            Element root = document.createElement("autokolcsonzo\_H2Z4X3");

            root.setAttribute("xmlns:xs", "http://www.w3.org/2001/XMLSchema-instance");

            root.setAttribute("xs:noNamespaceSchemaLocation", "XMLSchemaH2Z4X3.xsd");

            document.appendChild(root);

            // Telep

            root.appendChild(document.createComment("Telep"));

            addTelep(document, root, "1", "5", "Budapest", "Benczúr\_utca", "23", "B");

            addTelep(document, root, "2", "8", "Budapest", "Jókai\_utca", "14", "A");

            addTelep(document, root, "3", "6", "Szeged", "Szabadság\_utca", "23", "C");

            addTelep(document, root, "4", "6", "Siófok", "Balaton\_utca", "45", "A");

            // Tulajdonos

            root.appendChild(document.createComment("Tulajdonos"));

            addTulajdonos(document, root, "1", "1", "Kis János", "067022344555", "12345678-1-11","1500000" );

            addTulajdonos(document, root, "2", "2", "Cicz Imre", "06205678899", "18384763-2-24","1234560" );

            addTulajdonos(document, root, "3", "3", "Cset Elek", "06703452233", "2345679-3-13","1800000" );

            addTulajdonos(document, root, "4", "4", "Kis Anikó", "06304567788", "12341236-2-31","1850000" );

            // Autok

            root.appendChild(document.createComment("Autok"));

            addAuto(document, root, "1", "1", "PSY-213", "Jaguar\_F-TYPE", "JZMMA18P200411817","2020" );

            addAuto(document, root, "2", "2", "RYZ-022", "Mercedes\_S63\_AMG", "JZMMA18P200411817","2021" );

            addAuto(document, root, "3", "3", "SUV-123", "Volvo\_XC-90", "JZMMA18P201111111","2018");

            addAuto(document, root, "4", "4", "SRY-345", "Audi\_A5", "JZMMA18P200411817","2020" );

            addAuto(document, root, "5", "4", "PSY-234", "BMW\_M5\_Competition", "PZMMA18P200434567","2018" );

            // Berlo

            root.appendChild(document.createComment("Berlo"));

            addBerlo(document, root, "1", "2", "Diz Elek", "06703439988", "dizelek@gmail.com","WW001002" );

            addBerlo(document, root, "2", "3", "Farkas Piroska", "06703439988", "piroksa@gfreemail.hu","WW001002" );

            addBerlo(document, root, "3", "4", "Feles Elek", "06703439988", "felelek@gmail.com","MM020105" );

            addBerlo(document, root, "4", "5", "Git Áron", "06204563322", "gitaron@citromail.hu","WW001002" );

            // Alkalmazott

            root.appendChild(document.createComment("Akalmazott"));

            addAlkalmazott(document, root, "1", "2", "Hideg Ottó", "Adminisztrátor", "48","1975" );

            addAlkalmazott(document, root, "2", "3", "Kis Eszter", "Üzletvezető", "45","1978" );

            addAlkalmazott(document, root, "3", "1", "Pat Tamás", "supervisor", "35","1988" );

            addAlkalmazott(document, root, "4", "4", "Tra Pista", "marketing\_manager", "39","1984" );

            addAlkalmazott(document, root, "5", "2", "Tra Pista", "hr\_vezető", "45","1978" );

            // Bonusz

            root.appendChild(document.createComment("Bonusz"));

            addBonus(document, root, "1", "1", "1", "10000");

            addBonus(document, root, "2", "2", "2", "25000");

            addBonus(document, root, "3", "3", "3", "30000");

            // Kepesites

            root.appendChild(document.createComment("Képsítés"));

            addKepesites(document, root, "1",  "7245889", "2021","angol", "A2" );

            addKepesites(document, root, "2",  "7245867", "2015","magyar", "B2" );

            addKepesites(document, root, "3",  "7224589", "2010","angol", "C1" );

            addKepesites(document, root, "4",  "7298174", "2011","angol", "B1" );

            // Documentum felépítése

            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", "2");

            // Kimeneti fájl

            DOMSource source = new DOMSource(document);

            File outputFile = new File("XmlH2Z4X3\_out.xml");

            StreamResult file = new StreamResult(outputFile);

            transformer.transform(source, file);

            //Konzolra kiírás

            StreamResult console = new StreamResult(System.out);

            transformer.transform(source, console);

        }catch(Exception e) {

            e.printStackTrace();

        }

    }

    public static void addTelep(Document document, Element root, String id, String kapacitas, String varos,

            String utca, String hazszam, String minosites) {

        // Létrehoz egy "telep" elemet a megadott paraméterekkel.

        Element carstation = document.createElement("telep");

        carstation.setAttribute("TELEP\_ID", id);

        // Létrehoz egy "kapacitás" elemet és hozzáadja az értéket.

        Element capacity = createElementValue(document, "kapacitás", kapacitas);

        // Létrehoz egy "cim" elemet és hozzáadja az "irányítószám", "város", "utca\_házszám" elemeket.

        Element address = document.createElement("cim");

        Element city = createElementValue(document, "irányítószám", varos);

        Element street = createElementValue(document, "város", utca);

        Element number = createElementValue(document, "utca\_házszám", hazszam);

        Element quality = createElementValue(document, "mennyiség", minosites);

        // Hozzáadja az elemeket a megfelelő hierarchiában az "telep" elemhez.

        address.appendChild(city);

        address.appendChild(street);

        address.appendChild(number);

        carstation.appendChild(capacity);

        carstation.appendChild(address);

        carstation.appendChild(quality);

         // Hozzáadja az "telep" elemet a gyökérelemhez.

        root.appendChild(carstation);

    }

    public static void addTulajdonos(Document document, Element root, String id, String fk, String nev,

            String telszam, String adoszam, String bevetel) {

        Element owners = document.createElement("tulajdonos");

        owners.setAttribute("TELEP\_ID", id);

        owners.setAttribute("TUL\_TELEP", fk);

        // Elem létrehozása  a megadott paraméterekkel.

        Element name = createElementValue(document, "név", nev);

        Element phone = createElementValue(document, "telszám", telszam);

        Element taxid = createElementValue(document, "adoszám", adoszam);

        Element income = createElementValue(document, "bevétel", bevetel);

        owners.appendChild(name);

        owners.appendChild(phone);

        owners.appendChild(taxid);

        owners.appendChild(income);

        root.appendChild(owners);

    }

    public static void addAuto(Document document, Element root, String id, String fk, String rendszam,

            String modell, String alvazszam, String gyartev) {

        Element cars = document.createElement("autok");

        cars.setAttribute("AUTO\_ID", id);

        cars.setAttribute("AUTO\_TELEP", fk);

        // Elem létrehozása  a megadott paraméterekkel.

        Element licence = createElementValue(document, "rendszám", rendszam);

        Element model = createElementValue(document, "modell", modell);

        Element framenumber = createElementValue(document, "alvázszám", alvazszam);

        Element buildyear = createElementValue(document, "gáyrtásiév", gyartev);

         // Hozzáadja az elemeket a megfelelő hierarchiában

        cars.appendChild(licence);

        cars.appendChild(model);

        cars.appendChild(framenumber);

        cars.appendChild(buildyear);

         // Hozzáadja az "telep" elemet a gyökérelemhez.

        root.appendChild(cars);

    }

    public static void addBerlo(Document document, Element root, String id, String fk, String p1,

            String p2, String p3, String p4) {

        Element cars = document.createElement("bérlő");

        cars.setAttribute("BÉRLŐ\_ID", id);

        cars.setAttribute("BÉRLŐ\_AUTO", fk);

        // Elem létrehozása  a megadott paraméterekkel.

        Element name = createElementValue(document, "név", p1);

        Element phone = createElementValue(document, "telszám", p2);

        Element email = createElementValue(document, "email", p3);

        Element income = createElementValue(document, "jogositvanyszam", p4);

         // Hozzáadja az elemeket a megfelelő hierarchiában

        cars.appendChild(name);

        cars.appendChild(phone);

        cars.appendChild(email);

        cars.appendChild(income);

         // Hozzáadja az "telep" elemet a gyökérelemhez.

        root.appendChild(cars);

    }

    public static void addAlkalmazott(Document document, Element root, String id, String fk, String p1,

            String p2, String p3, String p4) {

        Element cars = document.createElement("alkalmazottak");

        cars.setAttribute("AUTO\_ID", id);

        cars.setAttribute("AUTO\_TELEP", fk);

        // Elem létrehozása  a megadott paraméterekkel.

        Element value1 = createElementValue(document, "név", p1);

        Element value2 = createElementValue(document, "beosztás", p2);

        Element value3 = createElementValue(document, "életkor", p3);

        Element value4 = createElementValue(document, "születésiév", p4);

         // Hozzáadja az elemeket a megfelelő hierarchiában

        cars.appendChild(value1);

        cars.appendChild(value2);

        cars.appendChild(value3);

        cars.appendChild(value4);

         // Hozzáadja az "telep" elemet a gyökérelemhez.

        root.appendChild(cars);

    }

    public static void addBonus(Document document, Element root, String id, String div\_fk,

            String kepesites\_fk, String bonus) {

        Element extra = document.createElement("bonusz");

        extra.setAttribute("BONUSZ\_ID", id);

        extra.setAttribute("ALKALMAZOTT\_FK", div\_fk);

        extra.setAttribute("KÉPESÍTÉS\_FK", kepesites\_fk);

        // Elem létrehozása  a megadott paraméterekkel.

        Element \_date = createElementValue(document, "mikortól", bonus);

        extra.appendChild(\_date);

         // Hozzáadja az "telep" elemet a gyökérelemhez.

        root.appendChild(extra);

    }

    public static void addKepesites(Document document, Element root, String id,  String p1,

            String p2, String p3, String p4) {

        Element certificate = document.createElement("képsítés");

        certificate.setAttribute("AUTO\_ID", id);

        // Elem létrehozása  a megadott paraméterekkel.

        Element value1 = createElementValue(document, "kapacitás", p1);

        Element value2 = createElementValue(document, "kapacitás", p2);

        Element value3 = createElementValue(document, "kapacitás", p3);

        Element value4 = createElementValue(document, "kapacitás", p4);

         // Hozzáadja az elemeket a megfelelő hierarchiában

        certificate.appendChild(value1);

        certificate.appendChild(value2);

        certificate.appendChild(value3);

        certificate.appendChild(value4);

         // Hozzáadja az "telep" elemet a gyökérelemhez.

        root.appendChild(certificate);

    }

    public static Element createElementValue(Document doc, String name, String value) {

        Element elem = doc.createElement(name);

        elem.appendChild(doc.createTextNode(value));

        return elem;

    }

}