JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

LoL e-sport csapatok

Készítette: **Kovács Krisztián**

Neptunkód: **WIQPM2**

Dátum: 2023.12.04.

Tartalom

[**Bevezetés** 3](#_Toc152632516)

[A feladat leírása 3](#_Toc152632517)

[**1.** **feladat** 3](#_Toc152632518)

[1a) Az adatbázis ER modell tervezése 3](#_Toc152632519)

[1b) Az adatbázis konvertálása XDM modellre 4](#_Toc152632520)

[1c) Az XDM modell alapján XML dokumentum készítése 5](#_Toc152632521)

[1d) Az XML dokumentum alapján XMLSchema készítése 9](#_Toc152632522)

[**2.** **feladat** 14](#_Toc152632523)

[2a) adatolvasás 14](#_Toc152632524)

[2b) adatmódosítás 15](#_Toc152632525)

[2c) adatlekérdezés 18](#_Toc152632526)

[2d) adatírás 23](#_Toc152632527)

# **Bevezetés**

## A feladat leírása

Ez a jegyzőkönyv a féléves feladat dokumentációjára készült az adatkezelés XML-ben című tárgyból. A feladatot a Visual Studio Code környezetben készítettem.

A féléves feladatom során a League of Legends nevezetű játék e-sport részének csapatainak adatnyilvántartó rendszerét hoztam létre. Különböző csapatok adatait tartalmazza, mint például maga a csapat és attribútumai, játékosok és attribútumai, csapatok edzői, megnyert versenyei, illetve szponzorjai. A modell öt egyedet tartalmaz, 1:1, 1:N, N:N kapcsolatokat, valamint összetett és többértékű tulajdonságokat is.

Az ER modell tervezésénél pontosítom az egyedeket, kapcsolatokat, illetve a tulajdonságokat.

# **feladat**

## 1a) Az adatbázis ER modell tervezése

Egyedek: Csapatok, Játékosok, Szponzorok, Események, Edzők

Kapcsolatok: Csapat 1:N Játékosok, Csapat 1:1 Edzők, Csapat 1:N Események, Csapat N:N Szponzorok.

Tulajdonságok:

Csapatok: név (kulcs), nemzetiség, rang, régió

Játékosok: azonosító (kulcs), név, nemzetiség, aktív, rang, pozíció, karrier kezdete, életkor

Szponzorok: név (kulcs), típus, ország, tulajdonos (többértékű)

Edzők: azonosító (kulcs), név, mióta edző, korábbi csapat

Események: azonosító (kulcs), név, év, nyeremény, helyszín (összetett: ország, város, hely)

A képen diagram, rajz, sor, minta látható

Automatikusan generált leírás

## 1b) Az adatbázis konvertálása XDM modellre

Az ER modell alapján az XDM modellt szerkesztettem. Gyökérelemként megjelenik a LoL\_WIQPM2, egyedeket és azok tulajdonságait ellipszissel ábrázoljuk, a többértékű tulajdonságot dupla körvonalúval. Téglalap jelöli a szöveg tartalmazását, a kulcsokat rombusz jelöli, a szöveg aláhúzva, az idegen kulcsok pedig szaggatott vonallal vannak jelölve. Az idegenkulcsok mutatnak a hivatkozott kulcsra.

Az N:N kapcsolat miatt létrejön a Mióta szponzor elem.

A képen vázlat látható

Automatikusan generált leírás

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

Az XDM alapján létrejön az XML. Minden többszörösen előforduló példányból létrehoztam minimum hármat. A kulcsok és idegen kulcsok attribútumok lesznek. A gyerekelemek lesznek a tulajdonságok.

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

<LoL\_WIQPM2 xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"

xs:noNamespaceSchemaLocation="XMLSchemaWIQPM2.xsd">

    <!-- Szponzorok -->

    <szponzorok>

        <szponzor sz\_nev = "Hana Bank">

            <tipus>Bank</tipus>

            <sz\_orszag>Dél-Korea</sz\_orszag>

            <tulajdonos>Park Sung-ho</tulajdonos>

        </szponzor>

        <szponzor sz\_nev = "Twitch">

            <tipus>Streamingszolgáltatás</tipus>

            <sz\_orszag>USA</sz\_orszag>

            <tulajdonos>Twitch Interactive</tulajdonos>

        </szponzor>

        <szponzor sz\_nev = "BMW">

            <tipus>Járműipar</tipus>

            <sz\_orszag>Németország</sz\_orszag>

            <tulajdonos>Stefan Quandt</tulajdonos>

            <tulajdonos>Susanne Klatten</tulajdonos>

            <tulajdonos>BlackRock</tulajdonos>

        </szponzor>

    </szponzorok>

    <!-- Szp csapat-->

    <szp\_csapatok>

        <szp\_csapat szp\_csapat\_azonosito = "1" szp\_nev = "BMW" csapat\_nev = "G2">

            <miota\_szp>2018</miota\_szp>

        </szp\_csapat>

        <szp\_csapat szp\_csapat\_azonosito = "2" szp\_nev = "Twitch" csapat\_nev = "G2">

            <miota\_szp>2020</miota\_szp>

        </szp\_csapat>

        <szp\_csapat szp\_csapat\_azonosito = "3" szp\_nev = "Hana Bank" csapat\_nev = "T1">

            <miota\_szp>2022</miota\_szp>

        </szp\_csapat>

    </szp\_csapatok>

    <!-- Játékosok -->

    <jatekosok>

        <jatekos j\_azonosito = "1" csapat\_nev = "G2">

            <j\_nev>BrokenBlade</j\_nev>

            <j\_nemzetiseg>német</j\_nemzetiseg>

            <aktiv>Igen</aktiv>

            <j\_rang>77</j\_rang>

            <pozicio>Top</pozicio>

            <karrier\_kezdete>2016</karrier\_kezdete>

            <eletkor>23</eletkor>

        </jatekos>

        <jatekos j\_azonosito = "2" csapat\_nev = "G2">

            <j\_nev>Yike</j\_nev>

            <j\_nemzetiseg>svéd</j\_nemzetiseg>

            <aktiv>Igen</aktiv>

            <j\_rang>203</j\_rang>

            <pozicio>Jungle</pozicio>

            <karrier\_kezdete>2019</karrier\_kezdete>

            <eletkor>23</eletkor>

        </jatekos>

        <jatekos j\_azonosito = "3" csapat\_nev = "G2">

            <j\_nev>Caps</j\_nev>

            <j\_nemzetiseg>norvég</j\_nemzetiseg>

            <aktiv>Igen</aktiv>

            <j\_rang>21</j\_rang>

            <pozicio>Mid</pozicio>

            <karrier\_kezdete>2015</karrier\_kezdete>

            <eletkor>24</eletkor>

        </jatekos>

        <jatekos j\_azonosito = "4" csapat\_nev = "G2">

            <j\_nev>Hans Sama</j\_nev>

            <j\_nemzetiseg>francia</j\_nemzetiseg>

            <aktiv>Igen</aktiv>

            <j\_rang>83</j\_rang>

            <pozicio>Bottom</pozicio>

            <karrier\_kezdete>2014</karrier\_kezdete>

            <eletkor>24</eletkor>

        </jatekos>

        <jatekos j\_azonosito = "5" csapat\_nev = "G2">

            <j\_nev>Mikyx</j\_nev>

            <j\_nemzetiseg>szlovén</j\_nemzetiseg>

            <aktiv>Igen</aktiv>

            <j\_rang>31</j\_rang>

            <pozicio>Support</pozicio>

            <karrier\_kezdete>2014</karrier\_kezdete>

            <eletkor>25</eletkor>

        </jatekos>

        <jatekos j\_azonosito = "6" csapat\_nev = "T1">

            <j\_nev>Faker</j\_nev>

            <j\_nemzetiseg>dél koreai</j\_nemzetiseg>

            <aktiv>Igen</aktiv>

            <j\_rang>2</j\_rang>

            <pozicio>Mid</pozicio>

            <karrier\_kezdete>2013</karrier\_kezdete>

            <eletkor>27</eletkor>

        </jatekos>

    </jatekosok>

    <!-- Csapatok -->

    <csapatok>

        <csapat cs\_nev = "T1">

            <cs\_nemzetiseg>Dél-Korea</cs\_nemzetiseg>

            <cs\_rang>1</cs\_rang>

            <regio>LCK</regio>

        </csapat>

        <csapat cs\_nev = "G2">

            <cs\_nemzetiseg>Németország</cs\_nemzetiseg>

            <cs\_rang>6</cs\_rang>

            <regio>LEC</regio>

        </csapat>

        <csapat cs\_nev = "Mad Lions">

            <cs\_nemzetiseg>Spanyolország</cs\_nemzetiseg>

            <cs\_rang>16</cs\_rang>

            <regio>LEC</regio>

        </csapat>

        <csapat cs\_nev = "JD Gaming">

            <cs\_nemzetiseg>Kína</cs\_nemzetiseg>

            <cs\_rang>4</cs\_rang>

            <regio>LPL</regio>

        </csapat>

    </csapatok>

    <!-- Edzők -->

    <edzok>

        <edzo e\_azonosito = "1" csapat\_nev = "G2">

            <e\_nev>Dylan Falco</e\_nev>

            <miota>2021</miota>

            <korabbi\_csapat>Schalke 04</korabbi\_csapat>

        </edzo>

        <edzo e\_azonosito = "2" csapat\_nev = "T1">

            <e\_nev>kK0ma</e\_nev>

            <miota>2023</miota>

            <korabbi\_csapat>DWG Kia</korabbi\_csapat>

        </edzo>

        <edzo e\_azonosito = "3" csapat\_nev = "Mad Lions">

            <e\_nev>Mac</e\_nev>

            <miota>2019</miota>

            <korabbi\_csapat>Splyce</korabbi\_csapat>

        </edzo>

        <edzo e\_azonosito = "4" csapat\_nev = "JD Gaming">

            <e\_nev>Mafa</e\_nev>

            <miota>2023</miota>

            <korabbi\_csapat>Gen.G</korabbi\_csapat>

        </edzo>

    </edzok>

    <!-- Események -->

    <esemenyek>

        <esemeny es\_azonosito = "1" csapat\_nev = "G2">

            <es\_nev>LEC 2023 Season Finals</es\_nev>

            <ev>2023</ev>

            <nyeremeny>65000</nyeremeny>

            <helyszin>

                <es\_orszag>Franciaország</es\_orszag>

                <varos>Montpellier</varos>

                <hely>Sud de France Arena</hely>

            </helyszin>

        </esemeny>

        <esemeny es\_azonosito = "2" csapat\_nev = "JD Gaming">

            <es\_nev>MSI 2023</es\_nev>

            <ev>2023</ev>

            <nyeremeny>50000</nyeremeny>

            <helyszin>

                <es\_orszag>Anglia</es\_orszag>

                <varos>London</varos>

                <hely>Queen Elizabeth Olympic Park</hely>

            </helyszin>

        </esemeny>

        <esemeny es\_azonosito = "3" csapat\_nev = "T1">

            <es\_nev>Worlds 2023</es\_nev>

            <ev>2023</ev>

            <nyeremeny>445000</nyeremeny>

            <helyszin>

                <es\_orszag>Dél-Korea</es\_orszag>

                <varos>Seul</varos>

                <hely>Gocheok Sky Dome</hely>

            </helyszin>

        </esemeny>

    </esemenyek>

</LoL\_WIQPM2>

## 1d) Az XML dokumentum alapján XMLSchema készítése

Az XML Schema validálta a létrehozott XML dokumentumot. Kigyűjtöttem az egyszerű típusokat, melyekre később hivatkozni fogok, létrehoztam saját típusokat, amelyben megszorítások vannak az adatra, mint például a pozíció 5 féle lehet: top, mid, bottom, support, jungle. Komplex típusokat létrehoztam az egyedekre, felépítettem a sorrendiséget. Ezután következett az elsődleges kulcsok, idegen kulcsok, illetve az 1:1 kapcsolat megvalósítása.

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

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

<!-- Egyszerű típusok -->

<xs:element name="tipus" type="xs:string" />

<xs:element name="sz\_orszag" type="xs:string" />

<xs:element name="tulajdonos" type="xs:string" />

<xs:element name="miota\_szp" type="xs:string" />

<xs:element name="j\_nev" type="xs:string" />

<xs:element name="j\_nemzetiseg" type="xs:string" />

<xs:element name="aktiv" type="aktivTipus" />

<xs:element name="j\_rang" type="xs:string" />

<xs:element name="pozicio" type="pozicioTipus" />

<xs:element name="karrier\_kezdete" type="xs:gYear" />

<xs:element name="eletkor" type="xs:integer" />

<xs:element name="cs\_nemzetiseg" type="xs:string" />

<xs:element name="cs\_rang" type="xs:integer" />

<xs:element name="regio" type="xs:string" />

<xs:element name="e\_nev" type="xs:string" />

<xs:element name="miota" type="xs:gYear" />

<xs:element name="korabbi\_csapat" type="xs:string" />

<xs:element name="es\_nev" type="xs:string" />

<xs:element name="ev" type="xs:gYear" />

<xs:element name="nyeremeny" type="xs:integer" />

<xs:element name="es\_orszag" type="xs:string" />

<xs:element name="varos" type="xs:string" />

<xs:element name="hely" type="xs:string" />

 <!-- Saját típusok létrehozása  -->

<xs:simpleType name="aktivTipus">

    <xs:restriction base="xs:string">

        <xs:enumeration value="Igen" />

        <xs:enumeration value="Nem" />

    </xs:restriction>

</xs:simpleType>

<xs:simpleType name="pozicioTipus">

    <xs:restriction base="xs:string">

        <xs:enumeration value="Top" />

        <xs:enumeration value="Mid" />

        <xs:enumeration value="Bottom" />

        <xs:enumeration value="Support" />

        <xs:enumeration value="Jungle" />

    </xs:restriction>

</xs:simpleType>

<!-- Komplex típushoz sajat típus létrehozása-->

<xs:complexType name="szponzorTipus">

    <xs:sequence>

        <xs:element ref="tipus" />

        <xs:element ref="sz\_orszag" />

        <xs:element name="tulajdonos" minOccurs="0" maxOccurs="unbounded"/>

    </xs:sequence>

    <xs:attribute name="sz\_nev" type="xs:string" use="required" />

</xs:complexType>

<xs:complexType name="szp\_csapatTipus">

    <xs:sequence>

        <xs:element ref="miota\_szp" />

    </xs:sequence>

    <xs:attribute name="szp\_csapat\_azonosito" type="xs:integer" use="required" />

    <xs:attribute name="szp\_nev" type="xs:string" use="required" />

    <xs:attribute name="csapat\_nev" type="xs:string" use="required" />

</xs:complexType>

<xs:complexType name="jatekosTipus">

    <xs:sequence>

        <xs:element ref="j\_nev" />

        <xs:element ref="j\_nemzetiseg" />

        <xs:element ref="aktiv" />

        <xs:element ref="j\_rang" />

        <xs:element ref="pozicio" />

        <xs:element ref="karrier\_kezdete" />

        <xs:element ref="eletkor" />

    </xs:sequence>

    <xs:attribute name="j\_azonosito" type="xs:integer" use="required" />

    <xs:attribute name="csapat\_nev" type="xs:string" use="required" />

</xs:complexType>

<xs:complexType name="csapatTipus">

    <xs:sequence>

        <xs:element ref="cs\_nemzetiseg" />

        <xs:element ref="cs\_rang" />

        <xs:element ref="regio" />

    </xs:sequence>

    <xs:attribute name="cs\_nev" type="xs:string" use="required" />

</xs:complexType>

<xs:complexType name="edzoTipus">

    <xs:sequence>

        <xs:element ref="e\_nev" />

        <xs:element ref="miota" />

        <xs:element ref="korabbi\_csapat" />

    </xs:sequence>

    <xs:attribute name="e\_azonosito" type="xs:integer" use="required" />

    <xs:attribute name="csapat\_nev" type="xs:string" use="required" />

</xs:complexType>

<xs:complexType name="esemenyTipus">

    <xs:sequence>

        <xs:element ref="es\_nev" />

        <xs:element ref="ev" />

        <xs:element ref="nyeremeny" />

        <xs:element name = "helyszin">

            <xs:complexType mixed = "true">

                <xs:sequence>

                    <xs:element name="es\_orszag" type="xs:string" />

                    <xs:element name="varos" type="xs:string" />

                    <xs:element name="hely" type="xs:string" />

                </xs:sequence>

            </xs:complexType>

        </xs:element>

    </xs:sequence>

    <xs:attribute name="es\_azonosito" type="xs:integer" use="required" />

    <xs:attribute name="csapat\_nev" type="xs:string" use="required" />

</xs:complexType>

<!-- Az elemek sorrendisége a gyökérelemtől -->

<xs:element name="LoL\_WIQPM2">

    <xs:complexType>

        <xs:sequence>

            <xs:element name="szponzorok">

                <xs:complexType mixed = "true">

                    <xs:sequence>

                        <xs:element name="szponzor" type="szponzorTipus" minOccurs="0" maxOccurs="unbounded"/>

                    </xs:sequence>

                </xs:complexType>

            </xs:element>

            <xs:element name="szp\_csapatok">

                <xs:complexType mixed = "true">

                    <xs:sequence>

                        <xs:element name="szp\_csapat" type="szp\_csapatTipus" minOccurs="0" maxOccurs="unbounded"/>

                    </xs:sequence>

                </xs:complexType>

            </xs:element>

            <xs:element name="jatekosok">

                <xs:complexType mixed = "true">

                    <xs:sequence>

                        <xs:element name="jatekos" type="jatekosTipus" minOccurs="0" maxOccurs="unbounded"/>

                    </xs:sequence>

                </xs:complexType>

            </xs:element>

            <xs:element name="csapatok">

                <xs:complexType mixed = "true">

                    <xs:sequence>

                        <xs:element name="csapat" type="csapatTipus" minOccurs="0" maxOccurs="unbounded"/>

                    </xs:sequence>

                </xs:complexType>

            </xs:element>

            <xs:element name="edzok">

                <xs:complexType mixed = "true">

                    <xs:sequence>

                        <xs:element name="edzo" type="edzoTipus" minOccurs="0" maxOccurs="unbounded"/>

                    </xs:sequence>

                </xs:complexType>

            </xs:element>

            <xs:element name="esemenyek">

                <xs:complexType mixed = "true">

                    <xs:sequence>

                        <xs:element name="esemeny" type="esemenyTipus" minOccurs="0" maxOccurs="unbounded"/>

                    </xs:sequence>

                </xs:complexType>

            </xs:element>

        </xs:sequence>

    </xs:complexType>

    <!-- Elsődleges kulcsok létrehozása-->

    <xs:key name="szponzor\_kulcs">

        <xs:selector xpath="szponzor"/>

        <xs:field xpath="@sz\_nev"/>

    </xs:key>

    <xs:key name="szp\_csapat\_kulcs">

        <xs:selector xpath="szp\_csapat"/>

        <xs:field xpath="@szp\_csapat\_azonosito"/>

    </xs:key>

    <xs:key name="jatekos\_kulcs">

        <xs:selector xpath="jatekos"/>

        <xs:field xpath="@j\_azonosito"/>

    </xs:key>

    <xs:key name="csapat\_kulcs">

        <xs:selector xpath="csapat"/>

        <xs:field xpath="@cs\_nev"/>

    </xs:key>

    <xs:key name="edzo\_kulcs">

        <xs:selector xpath="edzo"/>

        <xs:field xpath="@e\_azonosito"/>

    </xs:key>

    <xs:key name="esemeny\_kulcs">

        <xs:selector xpath="esemeny"/>

        <xs:field xpath="@es\_azonosito"/>

    </xs:key>

    <!-- Idegen kulcsok létrehozása -->

    <xs:keyref name="szp\_csapat\_szp\_ikulcs" refer="szponzor\_kulcs">

        <xs:selector xpath="szp\_csapat"/>

        <xs:field xpath="@szponzor"/>

    </xs:keyref>

    <xs:keyref name="szp\_csapat\_csapat\_ikulcs" refer="csapat\_kulcs">

        <xs:selector xpath="szp\_csapat"/>

        <xs:field xpath="@csapat"/>

    </xs:keyref>

    <xs:keyref name="jatekos\_csapat\_ikulcs" refer="csapat\_kulcs">

        <xs:selector xpath="jatekos"/>

        <xs:field xpath="@csapat"/>

    </xs:keyref>

    <xs:keyref name="edzo\_csapat\_ikulcs" refer="csapat\_kulcs">

        <xs:selector xpath="edzo"/>

        <xs:field xpath="@csapat"/>

    </xs:keyref>

    <xs:keyref name="esemeny\_csapat\_ikulcs" refer="esemeny\_kulcs">

        <xs:selector xpath="esemeny"/>

        <xs:field xpath="@csapat"/>

    </xs:keyref>

    <!-- 1:1 kapcsolat -->

    <xs:unique name="csapat\_edzo\_egyedi">

        <xs:selector xpath="edzo"/>

        <xs:field xpath="@csapat"/>

    </xs:unique>

</xs:element>

</xs:schema>

# **feladat**

## 2a) adatolvasás

Beolvasom a fájl tartalmát és a printNode metódus segítségével kiírom konzolra a teljes tartalmat megformázva.

package hu.domparse.wiqpm2;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.w3c.dom.Element;

import org.w3c.dom.NamedNodeMap;

public class DomReadWIQPM2 {

    public static void main(String[] args) {

        try {

                File xmlf= new File("XMLWIQPM2.xml");

                DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

                DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

                Document document = dBuilder.parse(xmlf);

                document.getDocumentElement().normalize();

                //Fa struktúra kiírása

                printNode(document.getDocumentElement(), 0);

            }

            catch (Exception e)

            {

                e.printStackTrace();

            }

        }

    private static void printNode(Node node, int depth)

    {

        String indent = " ".repeat(depth \* 5);

        Integer s = 0;

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

        {

            Element element = (Element) node;

            NamedNodeMap attributes = element.getAttributes();

            if(attributes.getLength() == 0){

                if(node.getChildNodes().getLength() > 1){

                    System.out.println(indent + "<" + node.getNodeName() + ">");

                    s = 1;

                }

                else{

                    System.out.print(indent + "<" + node.getNodeName() + ">");

                }

            }

            else{

                if(node.getChildNodes().getLength() > 1){

                    s = 1;

                }

                System.out.print(indent + "<" + node.getNodeName() + " ");

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

                    Node attribute = attributes.item(i);

                    System.out.print(attribute + " ");

                }

                if(node.getChildNodes().getLength() > 1){

                    System.out.println(">");

                }

                else{

                    System.out.print(">");

                }

            }

            NodeList nodeL = node.getChildNodes();

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

            {

                printNode(nodeL.item(i), depth + 1);

            }

            if(s == 1){

                System.out.print(indent + "</" + node.getNodeName() + ">");

                System.out.println();

            }

            else{

                System.out.print(" </" + node.getNodeName() + ">");

                System.out.println();

            }

        }

        else if (node.getNodeType() == Node.TEXT\_NODE && !node.getTextContent().trim().isEmpty())

        {

            System.out.print(" " +node.getTextContent().trim());

        }

    }

}

## 2b) adatmódosítás

Öt adatmódosítást végeztem, kommentben leírva, hogy mit módosít. A végén konzolra kiíratva leellenőriztem a végeredményt.

package hu.domparse.wiqpm2;

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.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

public class DomModifyWIQPM2 {

    public static void main(String[] args) {

        try {

            //xml file megnyitása

            File xmlf = new File("XMLWIQPM2.xml");

            //dokumentum létrehozasa és normalizálása

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

            Document document = documentBuilder.parse(xmlf);

            document.getDocumentElement().normalize();

            //metodusok

            query1(document);

            query2(document);

            query3(document);

            query4(document);

            query5(document);

            //a módosított adatok kiírasa a konzolra

            TransformerFactory transformerFactory = TransformerFactory.newInstance();

            Transformer transformer = transformerFactory.newTransformer();

            DOMSource source = new DOMSource(document);

            System.out.println("Módosítás után:\n");

            StreamResult consoleResult = new StreamResult(System.out);

            transformer.transform(source, consoleResult);

        } catch (Exception e ) {

            e.printStackTrace();

        }

    }

    public static void query1(Document document){

        // 1. Caps nevű játékos rangjának a módosítása

        NodeList nList = document.getElementsByTagName("jatekos");

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

            Node nNode = nList.item(i);

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

                Element elem = (Element) nNode;

                Node node1 = elem.getElementsByTagName("j\_nev").item(0);

                String text1 = node1.getTextContent();

                if("Caps".equals(text1)){

                    Node node2 = elem.getElementsByTagName("j\_rang").item(0);

                    node2.setTextContent("5");

                }

            }

        }

    }

    public static void query2(Document document){

        // 2. 3. id-jű játékos csapatának változtatása

        NodeList nList = document.getElementsByTagName("jatekos");

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

            Node nNode = nList.item(i);

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

                NamedNodeMap attr = nNode.getAttributes();

                Node nodeAttrID = attr.getNamedItem("j\_azonosito");

                Node nodeAttrCsapat = attr.getNamedItem("csapat\_nev");

                if (nodeAttrID.getTextContent().equals("3")) {

                    nodeAttrCsapat.setTextContent("Mad Lions");

                }

            }

        }

    }

    public static void query3(Document document){

        // 3. Szponzorált csapat modosítása

        NodeList nList = document.getElementsByTagName("szp\_csapat");

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

            Node nNode = nList.item(i);

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

                NamedNodeMap attribute = nNode.getAttributes();

                Node nodeAttrCsapat = attribute.getNamedItem("csapat\_nev");

                if (nodeAttrCsapat.getTextContent().equals("G2")) {

                    nodeAttrCsapat.setTextContent("T1");

                    Element elem = (Element) nNode;

                    Node node1 = elem.getElementsByTagName("miota\_szp").item(0);

                    node1.setTextContent("2023");

                }

            }

        }

    }

    public static void query4(Document document){

        // 4. Hana Bank nevű szponzor nevének változtatása

        NodeList nList = document.getElementsByTagName("szponzor");

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

            Node nNode = nList.item(i);

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

                Element elem = (Element) nNode;

                String sz\_nev = elem.getAttribute("sz\_nev");

                if (sz\_nev.equals("Hana Bank")) {

                    elem.setAttribute("sz\_nev", "Anah Bank");

                }

            }

        }

    }

    public static void query5(Document document){

        // 5. LEC régiót képviselő csapatok régiójának változtatása LCL-re

        NodeList nList = document.getElementsByTagName("csapat");

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

            Node nNode = nList.item(i);

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

                Element elem = (Element) nNode;

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

                String text1 = regio.getTextContent();

                if (text1.equals("LEC")) {

                    regio.setTextContent("LCL");

                }

            }

        }

    }

}

## 2c) adatlekérdezés

Öt lekérdezést készítettem, kommentben a lekérdezés magyarázata megtalálható. A konzolra írja ki a lekérdezések eredményét.

package hu.domparse.wiqpm2;

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.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

public class DomQueryWIQPM2 {

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

        try {

            //xml file megnyitasa

            File xmlf = new File("XMLWIQPM2.xml");

            //dokumentum létrehozása és normalizálása

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

            Document document = documentBuilder.parse(xmlf);

            document.getDocumentElement().normalize();

            //metodusok

            lekerdezes1(document);

            lekerdezes2(document);

            lekerdezes3(document);

            lekerdezes4(document);

            lekerdezes5(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. A LEC régiót képviselő csapatok:\n");

            NodeList nList = document.getElementsByTagName("csapat");

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

                Node nNode = nList.item(i);

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

                    Element elem = (Element) nNode;

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

                    String text1 = node1.getTextContent();

                    if("LEC".equals(text1)){

                        String csapat\_nev = elem.getAttribute("cs\_nev");

                        System.out.println("Csapat neve: " + csapat\_nev);

                        System.out.println("");

                    }

                }

            }

        } catch (NullPointerException e) {

            e.printStackTrace();

        }

        System.out.println("");

        System.out.println("");

    }

    public static void lekerdezes2(Document document) {

        try {

            System.out.println("\n2. Azon játékosok nevei és csapata, akinek az életkora 23 vagy kevesebb:\n");

            NodeList nList = document.getElementsByTagName("jatekos");

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

                Node nNode = nList.item(i);

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

                    Element elem = (Element) nNode;

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

                    String text1 = node1.getTextContent();

                    if(Integer.valueOf(text1) <= 23){

                        Node node2 = elem.getElementsByTagName("j\_nev").item(0);

                        String text2 = node2.getTextContent();

                        String csapat\_nev = elem.getAttribute("csapat\_nev");

                        System.out.println("Játékos neve: " + text2);

                        System.out.println("Csapat neve: " + csapat\_nev);

                        System.out.println("");

                    }

                }

            }

        } catch (NullPointerException e) {

            e.printStackTrace();

        }

        System.out.println("");

        System.out.println("");

    }

    public static void lekerdezes3(Document document) {

        try {

            System.out.println("\n3. Azon csapatok neve és rangja, amelyek megnyertek egy 100.000-nál nagyobb díjazású versenyt, illetve az esemény neve:\n");

            NodeList nList = document.getElementsByTagName("esemeny");

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

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

                Node nNode = nList.item(i);

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

                    Element elem = (Element) nNode;

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

                    String text1 = nyeremeny.getTextContent();

                    System.out.println(text1);

                    if(Integer.valueOf(text1) >= 100000){

                        String csapat\_nev = elem.getAttribute("csapat\_nev");

                        Node esemeny\_neve = elem.getElementsByTagName("es\_nev").item(0);

                        String text2 = esemeny\_neve.getTextContent();

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

                            Node nNode2 = nList2.item(j);

                            if (nNode2.getNodeType()==Node.ELEMENT\_NODE) {

                                Element elem2 = (Element) nNode2;

                                if(csapat\_nev.equals(elem2.getAttribute("cs\_nev"))){

                                    Node rang = elem2.getElementsByTagName("cs\_rang").item(0);

                                    String text3 = rang.getTextContent();

                                    System.out.println("Esemény: " + text2);

                                    System.out.println("Csapat neve: " + csapat\_nev);

                                    System.out.println("Rang: " + text3);

                                    System.out.println("");

                                }

                            }

                        }

                    }

                }

            }

        } catch (NullPointerException e) {

            e.printStackTrace();

        }

        System.out.println("");

        System.out.println("");

    }

    public static void lekerdezes4(Document document) {

        try {

            System.out.println("\n4. T1 csapat által nyert események:\n");

            NodeList nList = document.getElementsByTagName("esemeny");

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

                Node nNode = nList.item(i);

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

                    Element elem = (Element) nNode;

                    String csapat\_nev = elem.getAttribute("csapat\_nev");

                    if("T1".equals(csapat\_nev)){

                        Node esemeny\_neve = elem.getElementsByTagName("es\_nev").item(0);

                        String text1 = esemeny\_neve.getTextContent();

                        System.out.println("Esemény neve: " + text1);

                        System.out.println("");

                    }

                }

            }

        } catch (NullPointerException e) {

            e.printStackTrace();

        }

        System.out.println("");

        System.out.println("");

    }

    public static void lekerdezes5(Document document) {

        try {

            System.out.println("\n5. A G2 csapat játékosainak adatai:\n");

            NodeList nList = document.getElementsByTagName("jatekos");

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

                Node nNode = nList.item(i);

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

                    Element elem = (Element) nNode;

                    String csapat\_nev = elem.getAttribute("csapat\_nev");

                    if("G2".equals(csapat\_nev)){

                        String azonosito = elem.getAttribute("j\_azonosito");

                        System.out.println("Azonosító: " + azonosito);

                        System.out.println("Csapat neve: " + csapat\_nev);

                        NodeList nList2 = nNode.getChildNodes();

                        for(int j = 1; j < nList2.getLength(); j++){

                            Node nNode2 = nList2.item(j);

                            if (nNode2.getNodeType()==Node.ELEMENT\_NODE) {

                                Element elem2 = (Element) nNode2;

                                String text2 = elem2.getTextContent();

                                System.out.print(elem2.getNodeName() + ": " + text2 + ", ");

                            }

                        }

                        System.out.println("");

                    }

                }

            }

        } catch (NullPointerException e) {

            e.printStackTrace();

        }

        System.out.println("");

        System.out.println("");

    }

}

## 2d) adatírás

Egy újdokumentumot hozok létre, amelyet feltöltök adatokkal, ezután konzolra kiíratom azt.

package hu.domparse.wiqpm2;

import java.io.File;

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 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.Element;

import org.w3c.dom.NamedNodeMap;

public class DomWriteWIQPM2 {

    public static void main(String[] args) {

        try {

                DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

                DocumentBuilder builder = factory.newDocumentBuilder();

                Document document = builder.newDocument();

                fill(document);

                //Fa struktúra kiírása

                printNode(document.getDocumentElement(), 0);

                TransformerFactory transformerFactory = TransformerFactory.newInstance();

                Transformer transformer = transformerFactory.newTransformer();

                transformer.setOutputProperty(OutputKeys.INDENT, "yes"); // Bekezdések hozzáadása

                transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "5"); // Indentálási mélység

                DOMSource source = new DOMSource(document);

                StreamResult result = new StreamResult(new File("XMLWIQPM21.xml"));

                transformer.transform(source, result);

            }

            catch (Exception e)

            {

                e.printStackTrace();

            }

        }

    private static void printNode(Node node, int depth)

    {

        String indent = " ".repeat(depth \* 5);

        Integer s = 0;

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

        {

            Element element = (Element) node;

            NamedNodeMap attributes = element.getAttributes();

            if(attributes.getLength() == 0){

                if(node.getChildNodes().getLength() > 1){

                    System.out.println(indent + "<" + node.getNodeName() + ">");

                    s = 1;

                }

                else{

                    System.out.print(indent + "<" + node.getNodeName() + ">");

                }

            }

            else{

                if(node.getChildNodes().getLength() > 1){

                    s = 1;

                }

                System.out.print(indent + "<" + node.getNodeName() + " ");

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

                    Node attribute = attributes.item(i);

                    System.out.print(attribute + " ");

                }

                if(node.getChildNodes().getLength() > 1){

                    System.out.println(">");

                }

                else{

                    System.out.print(">");

                }

            }

            NodeList nodeL = node.getChildNodes();

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

            {

                printNode(nodeL.item(i), depth + 1);

            }

            if(s == 1){

                System.out.print(indent + "</" + node.getNodeName() + ">");

                System.out.println();

            }

            else{

                System.out.print(" </" + node.getNodeName() + ">");

                System.out.println();

            }

        }

        else if (node.getNodeType() == Node.TEXT\_NODE && !node.getTextContent().trim().isEmpty())

        {

            System.out.print(" " +node.getTextContent().trim());

        }

    }

    private static void fill(Document doc) {

        Element rootElement = doc.createElement("LoL\_WIQPM2");

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

        rootElement.setAttribute("xs:noNamespaceSchemaLocation", "XMLSchemaWIQPM2.xsd");

        doc.appendChild(rootElement);

        Element szponzorok = doc.createElement("szponzorok");

        addSzponzorok(doc, szponzorok, "Monster Energy", "Energiaital", "USA", "Rodney Sacks", null, null);

        addSzponzorok(doc, szponzorok, "SteelSeries", "Gaming", "Norvégia", "Ehtisham Rabbani", "Tino Soelberg", "Oluf Riddersholm");

        addSzponzorok(doc, szponzorok, "Twitch", "Streamingszolgáltatás", "USA", "Twitch Interactive", null, null);

        rootElement.appendChild(szponzorok);

        Element szp\_csapatok = doc.createElement("szp\_csapatok");

        addSzp\_csapatok(doc, szp\_csapatok, "1", "Monster Energy", "Fnatic", "2017");

        addSzp\_csapatok(doc, szp\_csapatok, "2", "SteelSeries", "Team Liquid", "2018");

        addSzp\_csapatok(doc, szp\_csapatok, "3", "Twitch", "PSG Talon", "2020");

        rootElement.appendChild(szp\_csapatok);

        Element jatekosok = doc.createElement("jatekosok");

        addJatekosok(doc, jatekosok, "1", "Fnatic", "Humanoid", "cseh", "Igen", "62", "Mid", "2016", "23");

        addJatekosok(doc, jatekosok, "2", "Fnatic", "Razork", "spanyol", "Igen", "167", "Jungle", "2016", "23");

        addJatekosok(doc, jatekosok, "3", "Team Liquid", "CoreJJ", "dél koreai", "Igen", "69", "Support", "2013", "29");

        rootElement.appendChild(jatekosok);

        Element csapatok = doc.createElement("csapatok");

        addCsapatok(doc, csapatok, "Fnatic", "Anglia", "14", "EMEA");

        addCsapatok(doc, csapatok, "Team Liquid", "Hollandia", "15", "NA");

        addCsapatok(doc, csapatok, "PSG Talon", "Kína", "22", "PCS");

        rootElement.appendChild(csapatok);

        Element edzok = doc.createElement("edzok");

        addEdzok(doc, edzok, "1", "Fnatic", "Nightshare", "2023", "Immortals");

        addEdzok(doc, edzok, "2", "Team Liquid", "Spawn", "2023", "TL Challengers");

        addEdzok(doc, edzok, "3", "PSG Talon", "CorGi", "2021", "EDward Gaming");

        rootElement.appendChild(edzok);

        Element esemenyek = doc.createElement("esemenyek");

        addEsemenyek(doc, esemenyek, "1", "Fnatic", "Rift Rivals 2018 NA-EU", "2018", "20000", "US", "Los Angeles", "Riot North American Studios");

        addEsemenyek(doc, esemenyek, "2", "Fnatic", "EU LCS 2015 Summer Playoffs", "2015", "50000", "Svédország", "Stockholm", "Hovet Arena");

        addEsemenyek(doc, esemenyek, "3", "Team Liquid", " LCS 2022 Lock In", "2022", "150000", "US", "Los Angeles", "LCS Studio");

        rootElement.appendChild(esemenyek);

    }

    private static void addSzponzorok(Document doc, Element parentElement, String sz\_nev, String tipus, String sz\_orszag, String tulajdonos1, String tulajdonos2, String tulajdonos3) {

        Element szponzorE = doc.createElement("szponzor");

        szponzorE.setAttribute("sz\_nev", sz\_nev);

        Element tipusE = createElement(doc, "tipus", tipus);

        szponzorE.appendChild(tipusE);

        Element sz\_orszagE = createElement(doc, "sz\_orszag", sz\_orszag);

        szponzorE.appendChild(sz\_orszagE);

        if (tulajdonos1 != null && !tulajdonos1.isEmpty()) {

            Element tulajdonos1E = createElement(doc, "tulajdonos", tulajdonos1);

            szponzorE.appendChild(tulajdonos1E);

        }

        if (tulajdonos2 != null && !tulajdonos2.isEmpty()) {

            Element tulajdonos2E = createElement(doc, "tulajdonos", tulajdonos2);

            szponzorE.appendChild(tulajdonos2E);

        }

        if (tulajdonos3 != null && !tulajdonos3.isEmpty()) {

            Element tulajdonos3E = createElement(doc, "tulajdonos", tulajdonos3);

            szponzorE.appendChild(tulajdonos3E);

        }

        parentElement.appendChild(szponzorE);

    }

    private static void addSzp\_csapatok(Document doc, Element parentElement, String szp\_csapat\_azonosito, String szp\_nev, String csapat\_nev, String miota\_szp) {

        Element szponzoralt\_csE = doc.createElement("szp\_csapat");

        szponzoralt\_csE.setAttribute("szp\_csapat\_azonosito", szp\_csapat\_azonosito);

        szponzoralt\_csE.setAttribute("szp\_nev", szp\_nev);

        szponzoralt\_csE.setAttribute("csapat\_nev", csapat\_nev);

        Element miota\_szpE = createElement(doc, "miota\_szp", miota\_szp);

        szponzoralt\_csE.appendChild(miota\_szpE);

        parentElement.appendChild(szponzoralt\_csE);

    }

    private static void addJatekosok(Document doc, Element parentElement, String j\_azonosito, String csapat\_nev, String j\_nev, String j\_nemzetiseg, String aktiv, String j\_rang, String pozicio, String karrier\_kezdete, String eletkor) {

        Element jatekosE = doc.createElement("jatekos");

        jatekosE.setAttribute("j\_azonosito", j\_azonosito);

        jatekosE.setAttribute("csapat\_nev", csapat\_nev);

        Element j\_nevE = createElement(doc, "j\_nev", j\_nev);

        jatekosE.appendChild(j\_nevE);

        Element j\_nemzetisegE = createElement(doc, "j\_nemzetiseg", j\_nemzetiseg);

        jatekosE.appendChild(j\_nemzetisegE);

        Element aktivE = createElement(doc, "aktiv", aktiv);

        jatekosE.appendChild(aktivE);

        Element j\_rangE = createElement(doc, "j\_rang", j\_rang);

        jatekosE.appendChild(j\_rangE);

        Element pozicioE = createElement(doc, "pozicio", pozicio);

        jatekosE.appendChild(pozicioE);

        Element karrier\_kezdeteE = createElement(doc, "karrier\_kezdete", karrier\_kezdete);

        jatekosE.appendChild(karrier\_kezdeteE);

        Element eletkorE = createElement(doc, "eletkor", eletkor);

        jatekosE.appendChild(eletkorE);

        parentElement.appendChild(jatekosE);

    }

    private static void addCsapatok(Document doc, Element parentElement, String cs\_nev, String cs\_nemzetiseg, String cs\_rang, String regio) {

        Element csapatE = doc.createElement("csapat");

        csapatE.setAttribute("cs\_nev", cs\_nev);

        Element cs\_nemzetisegE = createElement(doc, "cs\_nemzetiseg", cs\_nemzetiseg);

        csapatE.appendChild(cs\_nemzetisegE);

        Element cs\_rangE = createElement(doc, "cs\_rang", cs\_rang);

        csapatE.appendChild(cs\_rangE);

        Element regioE = createElement(doc, "regio", regio);

        csapatE.appendChild(regioE);

        parentElement.appendChild(csapatE);

    }

    private static void addEdzok(Document doc, Element parentElement, String e\_azonosito, String csapat\_nev, String e\_nev, String miota, String korabbi\_csapat) {

        Element edzoE = doc.createElement("edzo");

        edzoE.setAttribute("e\_azonosito", e\_azonosito);

        edzoE.setAttribute("csapat\_nev", csapat\_nev);

        Element e\_nevE = createElement(doc, "e\_nev", e\_nev);

        edzoE.appendChild(e\_nevE);

        Element miotaE = createElement(doc, "miota", miota);

        edzoE.appendChild(miotaE);

        Element korabbi\_csapatE = createElement(doc, "korabbi\_csapat", korabbi\_csapat);

        edzoE.appendChild(korabbi\_csapatE);

        parentElement.appendChild(edzoE);

    }

    private static void addEsemenyek(Document doc, Element parentElement, String es\_azonosito, String csapat\_nev, String es\_nev, String ev, String nyeremeny, String es\_orszag, String varos, String hely) {

        Element esemenyE = doc.createElement("esemeny");

        esemenyE.setAttribute("es\_azonosito", es\_azonosito);

        esemenyE.setAttribute("csapat\_nev", csapat\_nev);

        Element es\_nevE = createElement(doc, "es\_nev", es\_nev);

        esemenyE.appendChild(es\_nevE);

        Element evE = createElement(doc, "ev", ev);

        esemenyE.appendChild(evE);

        Element nyeremenyE = createElement(doc, "nyeremeny", nyeremeny);

        esemenyE.appendChild(nyeremenyE);

        Element helyszinE = doc.createElement("helyszin");

        Element es\_orszagE = createElement(doc, "es\_orszag", es\_orszag);

        helyszinE.appendChild(es\_orszagE);

        Element varosE = createElement(doc, "varos", varos);

        helyszinE.appendChild(varosE);

        Element helyE = createElement(doc, "hely", hely);

        helyszinE.appendChild(helyE);

        esemenyE.appendChild(helyszinE);

        parentElement.appendChild(esemenyE);

    }

    private static Element createElement(Document doc, String name, String value) {

        Element element = doc.createElement(name);

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

        return element;

    }

}