

JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Filmadatbázis

Készítette: **Garay Gabriel**

Neptunkód: **GJ2N7R**

Dátum: **2023.12.04**

Tartalomjegyzék

Bevezetés.....	3
A feladat leírása.....	3
1. feladat	5
a) Az adatbázis ER modell tervezése	5
b) Az adatbázis konvertálása XDM modellre	8
c) Az XDM modell alapján XML dokumentum készítése:.....	9
d) Az XML dokumentum alapján XMLSchema készítése	14
2. feladat	20
a) Adatolvasás	20
b) Adatmódosítás	24
c) Adatlekérdezés	29
d) Adatírás.....	36

Bevezetés

Munkám célja az XML és köré épülő technológiák elsajítása. Magába foglalja az XML dokumentum létrehozását, amit az ER és XDM modell előz meg, és az xsd séma követ.

A dokumentumban található adatok kezelésére Java programokat írtam, amiket igyekeztem érthetően és részletesen ellátni megjegyzésekkel.

Ehhez a Visual Studio Code fejlesztőkörnyezetet vettem igénybe.

A feladat leírása

A feladat elvégzéséhez egy **Filmadatbázis** nevű adatnyilvántartó rendszert hoztam létre. Feladatom középpontjában a **filmek** állnak, melyeknek *címük*, *kiadási évük* és *műfajaik* adottak. Ez utóbbiból több is jelen lehet egy filmnél, mint például a vígjáték, akció, horror és még 10 lehetséges opció. Az opciók a **kategórián** belül találhatók, *névvel* ellátva. Minden kategóriához egy rövid, pontosító *leírás* társul. A filmeket a **felhasználók** tekintik meg. Őket *felhasználónevükkel*, *születési dátumukkal* és *email címükkel* találjuk meg a rendszerben. Minden film kap egy **értékelést**, mely megadja a filmhez tartozó *pontszámot*, *hányan értékelték* a filmet, illetve, hogy az értékelésekhez milyen *szöveg* társult.

Az élszereplős filmeknél mindig találunk egy szereplőlistát azokról, akik a karaktereket eljátsszák, vagy akár animációs filmeknél narrálják. Ők lesznek a **színészek**. Nekik tároljuk a *nevüket*, *születési- dátumukat* és *helyüket*. Az elköteleződött színészekhez tartozhat egy **élettárs**, akivel együtt él. Élettársaknak ugyanazon tulajdonságok adhatók, mint a színészeknek. Ahhoz a színészhez, aki szerepléséért **díjat** kapott, tárolható annak neve (*típusként* van megadva a feladatban, de valójában ez egy név), a feltétel a díj elnyeréséért, illetve a díj korábbi nyerteseinek nevei.

1. feladat

a) Az adatbázis ER modell tervezése

Az ER modellben hét egyed található: film, felhasználó, értékelés, kategória, színész, élettárs és díj. Minden egyed rendelkezik egyedi, megkülönböztethető kulccsal, folytonos vonallal jelölve. Többértékű tulajdonság három egyed van: **értékelés**nél az *értékelés szövege*, **film**nél a hozzá tartozó *műfajok*, illetve **díj**nál a *nyertesek* nevei. Összetett tulajdonságból is van kettő a modellben, bár ezek azonosak: az **élettárs**nál és **színész**nél a *név* tulajdonság.

Három fajta kapcsolat látható a modellben:

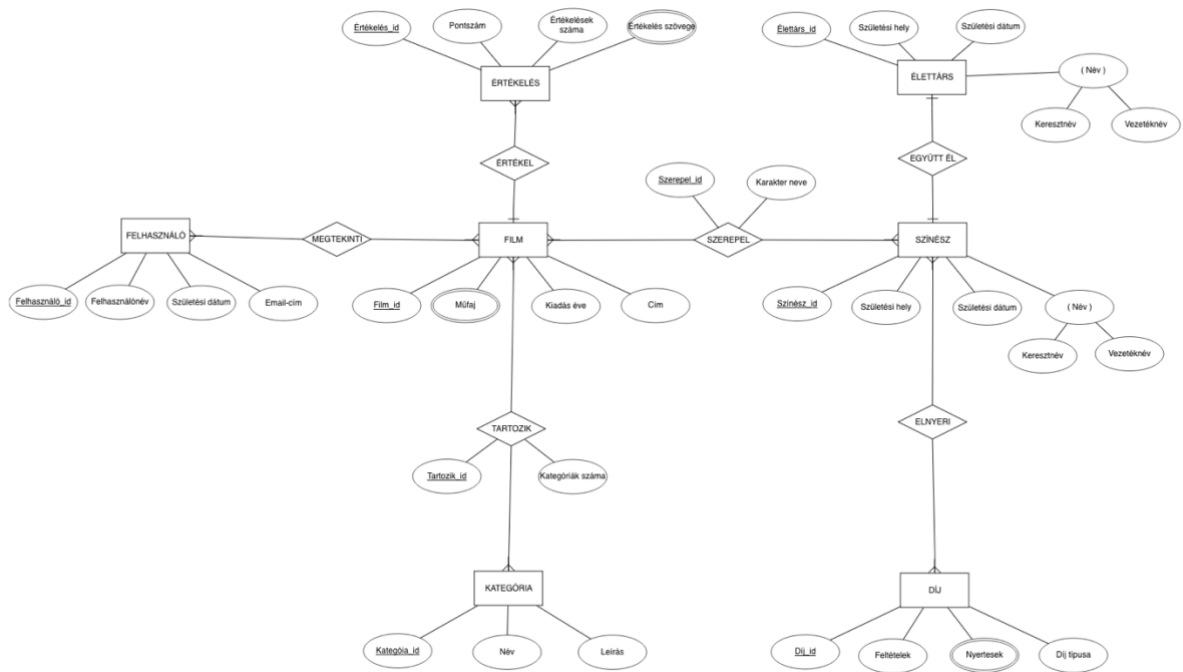
- 1:1 kapcsolat: A **színész** és **élettárs** között.
- 1:M kapcsolat: A **film** és **értékelés** között.
- N:M kapcsolat: A **film** és **felhasználó**, **film** és **kategória**, **film** és **színész**, valamint a **színész** és **díj** között.

A modell egyedei:

- Film
 - o Film_id: egyedi azonosító
 - o Műfaj: a filmet jellemző műfajok
 - o Kiadás éve: mikor jelent meg a film
 - o Cím: a film teljes neve
- Értékelés
 - o Értékelés_id: egyedi azonosító
 - o Pontszám: a film értékelése
 - o Értékelések száma: a filmet értékelők száma
 - o Értékelés szövege: értékelésekhez tartozó szöveg
- Felhasználó
 - o Felhasználó_id: egyedi azonosító
 - o Felhasználónév: a felhasználó által választott név
 - o Születési dátum: a felhasználó születési dátuma
 - o Email-cím: a felhasználó email-címe
- Kategória
 - o Kategória_id: egyedi azonosító
 - o Név: a kategória neve

- Leírás: a kategória rövid jellemzése
- Színész
 - Színész_id: egyedi azonosító
 - Születési hely: a színész születésének helye
 - Születési dátum: a színész születési dátuma
 - Név: A színész neve, mely összetett, további két tulajdonságból áll
 - Keresztnév
 - Vezetéknév
- Élettárs
 - Színész_id: egyedi azonosító
 - Születési hely: az élettársszületésének helye
 - Születési dátum: az élettárs születési dátuma
 - Név: Az élettárs neve, mely összetett, további két tulajdonságból áll
 - Keresztnév
 - Vezetéknév
- Díj
 - Díj_id: egyedi azonosító
 - Feltételek: a díj elnyerésének feltétele
 - Nyertesek: eddigi nyertesek felsorolása
 - Díj típusa: a díj neve
- Szerepel kapcsolat
 - Szerepel_id: egyedi azonosító
 - Karakter neve: a színész által eljátszott karakter neve
- Tartozik kapcsolat
 - Tartozik_id: egyedi azonosító
 - Kategóriák száma: a filmet leíró kategóriák száma

Az ER modell:



1.ábra: Az ER modell

b) Az adatbázis konvertálása XDM modellre

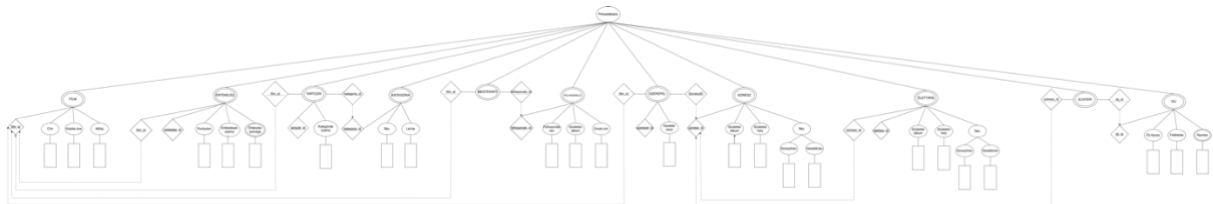
Az ER modell alapján elkészült XDM modell. A **Filmadatbázis** gyökérelemből indul, minden további elem a gyerekeleme, melyekből tíz darabot tartalmaz.

N:M kapcsolat megvalósításánál egy új elemet hoztam létre, mely két idegen kulcsot tartalmaz, amik a két megfelelő egyed elsődleges kulcsaira mutatnak. Ebből jött létre a **tartozik**, **megtekinti**, **szerepel** és az **elnyeri** elem.

1:M és 1:1 kapcsolatnál az egyik egyedhez szintén egy idegen kulcsot társítottam, ami a másik egyed elsődleges kulcsát célozza.

Összetett tulajdonság jelölésére a tulajdonsághoz a megfelelő attribútumokat társítottam, a színész és élettárs nevénél.

Az XDM modell: (mérete miatt nem látható élesen)



2.ábra: Az XDM modell

c) Az XDM modell alapján XML dokumentum készítése:

Az XML dokumentumban az XDM modell minden eleme egy XML elemet, vagy más néven példányt képez. Az elemek attribútumai egyedi azonosítókat, „id” -kat neveznek meg. Minden más gyerekelem az XML elemekben is gyerekelemként szerepelnek.

Kommentekkel láttam el minden példány kezdetét

Az XML dokumentum kódja:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Filmatadbázis xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
  xs:noNamespaceSchemaLocation="../xsd/XMLSchemaGJ2N7R.xsd">

  <!-- Film példányok -->
  <Film film_id="1">
    <Cím>Inception</Cím>
    <Kiadás_éve>2010</Kiadás_éve>
    <Műfaj>Sci-fi</Műfaj>
  </Film>

  <Film film_id="2">
    <Cím>The Shawshank Redemption</Cím>
    <Kiadás_éve>1994</Kiadás_éve>
    <Műfaj>Dráma</Műfaj>
  </Film>

  <Film film_id="3">
    <Cím>Avatar</Cím>
    <Kiadás_éve>2009</Kiadás_éve>
    <Műfaj>Sci-fi</Műfaj>
  </Film>

  <Film film_id="4">
    <Cím>The Dark Knight</Cím>
    <Kiadás_éve>2008</Kiadás_éve>
    <Műfaj>Akció</Műfaj>
  </Film>

  <!-- Értékelés példányok -->
  <Értékelés értékes_id="1" film_id="1">
    <Pontszám>4.8</Pontszám>
    <Értékelések_száma>1500</Értékelések_száma>
    <Értékelés_szövege>Nagyon jó film, érdemes megnézni!</Értékelés_szövege>
    <Értékelés_szövege>Lenyűgöző képi világ!</Értékelés_szövege>
    <Értékelés_szövege>Remek színészi alakítások!</Értékelés_szövege>
  </Értékelés>

  <Értékelés értékes_id="2" film_id="2">
    <Pontszám>4.9</Pontszám>
    <Értékelések_száma>2000</Értékelések_száma>
```

```

    <Értékelés_szövege>Minden idők egyik legjobb filmje!</Értékelés_szövege>
    <Értékelés_szövege>Nagyon izgalmas történet!</Értékelés_szövege>
  </Értékelés>
<Értékelés_ertekeles_id="3" film_id="3">
  <Pontszám>4.5</Pontszám>
  <Értékelések_száma>1200</Értékelések_száma>
  <Értékelés_szövege>Varázslatos filmélmény!</Értékelés_szövege>
  <Értékelés_szövege>Nagyszerű rendezés!</Értékelés_szövege>
  <Értékelés_szövege>Lenyűgöző látványvilág!</Értékelés_szövege>
</Értékelés>

<Értékelés_ertekeles_id="4" film_id="4">
  <Pontszám>4.7</Pontszám>
  <Értékelések_száma>1800</Értékelések_száma>
  <Értékelés_szövege>Nagyon élvezetes film!</Értékelés_szövege>
  <Értékelés_szövege>Izgalmas cselekmény!</Értékelés_szövege>
</Értékelés>

<!-- Kategória példányok -->
<Kategória_kategoria_id="1">
  <Kategória_név>Akció</Kategória_név>
  <Leírás>Izgalmas, pörgős jeleneteket tartalmazó filmek</Leírás>
</Kategória>

<Kategória_kategoria_id="2">
  <Kategória_név>Drama</Kategória_név>
  <Leírás>Mély érzelmekre épülő filmek</Leírás>
</Kategória>

<Kategória_kategoria_id="3">
  <Kategória_név>Sci-fi</Kategória_név>
  <Leírás>Fantázia és tudományos elemeket tartalmazó filmek</Leírás>
</Kategória>

<!-- Tartozik kapcsolótábla példányok -->
<Tartozik_film_id="1" kategoria_id="1" tartozik_id="1">
  <Kategóriák_száma>2</Kategóriák_száma>
</Tartozik>

<Tartozik_film_id="2" kategoria_id="2" tartozik_id="2">
  <Kategóriák_száma>1</Kategóriák_száma>
</Tartozik>

<Tartozik_film_id="3" kategoria_id="3" tartozik_id="3">
  <Kategóriák_száma>3</Kategóriák_száma>
</Tartozik>
<Tartozik_film_id="1" kategoria_id="1" tartozik_id="4">
  <Kategóriák_száma>2</Kategóriák_száma>
</Tartozik>

```

```

<!-- Felhasználó példányok -->
<Felhasználó felhasznalo_id="1">
  <Felhasználónév>user1</Felhasználónév>
  <Születési_dátum>1985-05-15</Születési_dátum>
  <Email-cím>user1@example.com</Email-cím>
</Felhasználó>

<Felhasználó felhasznalo_id="2">
  <Felhasználónév>user2</Felhasználónév>
  <Születési_dátum>1990-08-22</Születési_dátum>
  <Email-cím>user2@example.com</Email-cím>
</Felhasználó>

<Felhasználó felhasznalo_id="3">
  <Felhasználónév>user3</Felhasználónév>
  <Születési_dátum>1988-03-10</Születési_dátum>
  <Email-cím>user3@example.com</Email-cím>
</Felhasználó>

<Felhasználó felhasznalo_id="4">
  <Felhasználónév>user4</Felhasználónév>
  <Születési_dátum>1995-12-05</Születési_dátum>
  <Email-cím>user4@example.com</Email-cím>
</Felhasználó>

<!-- Megtekinti kapcsolótábla példányok -->
<Megtekinti felhasznalo_id="1" film_id="1" />
<Megtekinti felhasznalo_id="2" film_id="2" />
<Megtekinti felhasznalo_id="3" film_id="3" />
<Megtekinti felhasznalo_id="4" film_id="4" />

<!-- Színész példányok -->
<Színész szinesz_id="1">
  <Név>
    <Keresztnév>Leonardo</Keresztnév>
    <Vezetéknév>DiCaprio</Vezetéknév>
  </Név>
  <Születési_dátum>1974-11-11</Születési_dátum>
  <Születési_hely>Los Angeles, Kalifornia</Születési_hely>
</Színész>
<Színész szinesz_id="2">
  <Név>
    <Keresztnév>Morgan</Keresztnév>
    <Vezetéknév>Freeman</Vezetéknév>
  </Név>
  <Születési_dátum>1937-06-01</Születési_dátum>
  <Születési_hely>Memphis, Tennessee</Születési_hely>
</Színész>

<Színész szinesz_id="3">
  <Név>

```

```

    <Keresztnév>Sam</Keresztnév>
    <Vezetéknév>Worthington</Vezetéknév>
  </Név>
  <Születési_dátum>1976-08-02</Születési_dátum>
  <Születési_hely>Godalming, Egyesült Királyság</Születési_hely>
</Színész>

<!-- Szerepel példányok -->
<Szerepel film_id="1" szerepel_id="1" színesz_id="1">
  <Karakter_neve>Dominic Cobb</Karakter_neve>
</Szerepel>

<Szerepel film_id="2" szerepel_id="2" színesz_id="2">
  <Karakter_neve>Andy Dufresne</Karakter_neve>
</Szerepel>

<Szerepel film_id="3" szerepel_id="3" színesz_id="3">
  <Karakter_neve>Jake Sully</Karakter_neve>
</Szerepel>

<!-- Élettárs példányok -->
<Élettárs elettsars_id="1" színesz_id="1">
  <Név>
    <Keresztnév>Camila</Keresztnév>
    <Vezetéknév>Morrone</Vezetéknév>
  </Név>
  <Születési_dátum>1997-06-16</Születési_dátum>
  <Születési_hely>Buenos Aires, Argentina</Születési_hely>
</Élettárs>

<Élettárs elettsars_id="2" színesz_id="2">
  <Név>
    <Keresztnév>Myrna</Keresztnév>
    <Vezetéknév>Colley-Lee</Vezetéknév>
  </Név>
  <Születési_dátum>1941-03-15</Születési_dátum>
  <Születési_hely>Milwaukee, Wisconsin</Születési_hely>
</Élettárs>

<Élettárs elettsars_id="3" színesz_id="3">
  <Név>
    <Keresztnév>Lara</Keresztnév>
    <Vezetéknév>Worthington</Vezetéknév>
  </Név>
  <Születési_dátum>1976-08-02</Születési_dátum>
  <Születési_hely>Godalming, Egyesült Királyság</Születési_hely>
</Élettárs>

<!-- Díj példányok -->
<Díj díj_id="1">
  <Díj_típusa>Oscar</Díj_típusa>

```

```
<Feltételek>Legjobb film</Feltételek>
<Nyertes>Christopher Nolan</Nyertes>
<Nyertes>Matthew McConaughey</Nyertes>
</Díj>
```

```
<Díj díj_id="2">
  <Díj_típusa>Golden Globe</Díj_típusa>
  <Feltételek>Legjobb színész</Feltételek>
  <Nyertes>Leonardo DiCaprio</Nyertes>
  <Nyertes>Tom Hanks</Nyertes>
  <Nyertes>Emma Stone</Nyertes>
</Díj>
```

```
<Díj díj_id="3">
  <Díj_típusa>BAFTA</Díj_típusa>
  <Feltételek>Legjobb rendező</Feltételek>
  <Nyertes>Alfonso Cuarón</Nyertes>
  <Nyertes>Greta Gerwig</Nyertes>
</Díj>
```

```
<!-- Elnyeri kapcsolótábla példányok -->
<Elnyeri színesz_id="1" díj_id="1"></Elnyeri>
<Elnyeri színesz_id="2" díj_id="2"></Elnyeri>
<Elnyeri színesz_id="3" díj_id="3"></Elnyeri>
```

```
</Filmadatbázis>
```

d) Az XML dokumentum alapján XMLSchema készítése

Az xsd séma alapjául az XML dokumentum szolgált.

Először létrehoztam a **saját típusokat**, melyeket fel tudtam használni a komplex típusoknál. A *műfaj* tizenhárom lehetséges értéket kapott, a *pontszám* egy intervallumot 0-tól 10-ig, és mivel a filmek pontszámai gyakran tizedesszámok, ezért „float” típust használtam. A kategóriáknál pedig megszabtam, hogy egy filmhez legfeljebb öt kategória tartozhat.

Komplex típusokba tartozik minden egyed- és kapcsolat típusa, illetve az összetett **Név** típus a két egyszerű elemével.

Elsődleges kulcs lett minden egyedi azonosító, **idegen kulcs** pedig a kapcsolatokat létrehozó, elsődleges kulcsokra mutató azonosító.

Az xsd séma kódja:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- Saját típusok létrehozása-->
  <xs:simpleType name="MűfajType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Horror" />
      <xs:enumeration value="Sci-fi" />
      <xs:enumeration value="Dráma" />
      <xs:enumeration value="Romantikus" />
      <xs:enumeration value="Akció" />
      <xs:enumeration value="Thriller" />
      <xs:enumeration value="Musical" />
      <xs:enumeration value="Vígjáték" />
      <xs:enumeration value="Animáció" />
      <xs:enumeration value="Krimi" />
      <xs:enumeration value="Dokumentum" />
      <xs:enumeration value="Fantasy" />
      <xs:enumeration value="Történelmi" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="PontszámType">
    <xs:restriction base="xs:float">
      <xs:minInclusive value="0" />
      <xs:maxInclusive value="10" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="Kategóriák_számaType">
    <xs:restriction base="xs:int">
      <xs:minInclusive value="1" />
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

```

        <xs:maxInclusive value="5" />
    </xs:restriction>
</xs:simpleType>

<!-- Egyszerű típusok létrehozása-->
<xs:element name="Születési_dátum" type="xs:string" />
<xs:element name="Születési_hely" type="xs:string" />

<!-- Komplex típusok létrehozása-->
<xs:complexType name="NévType">
    <xs:sequence>
        <xs:element name="Keresztnev" type="xs:string" />
        <xs:element name="Vezetéknév" type="xs:string" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="FilmType">
    <xs:sequence>
        <xs:element name="Cím" type="xs:string" />
        <xs:element name="Kiadás_éve" type="xs:gYear" />
        <xs:element name="Műfaj" type="MűfajType" />
    </xs:sequence>
    <xs:attribute name="film_id" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="ÉrtékelésType">
    <xs:sequence>
        <xs:element name="Pontszám" type="PontszámType"/>
        <xs:element name="Értékelések_száma" type="xs:string"/>
        <xs:element name="Értékelés_szövege" type="xs:string" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="ertekeles_id" type="xs:integer" use="required" />
    <xs:attribute name="film_id" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="KategóriaType">
    <xs:sequence>
        <xs:element name="Kategória_név" type="xs:string"/>
        <xs:element name="Leírás" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="kategoria_id" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="TartozikType">
    <xs:sequence>
        <xs:element name="Kategóriák_száma" type="Kategóriák_számaType" />
    </xs:sequence>
    <xs:attribute name="tartozik_id" type="xs:integer" use="required" />
    <xs:attribute name="film_id" type="xs:integer" use="required" />
    <xs:attribute name="kategoria_id" type="xs:integer" use="required" />
</xs:complexType>

```

```

<xs:complexType name="FelhasználóType">
  <xs:sequence>
    <xs:element name="Felhasználónév" type="xs:string" />
    <xs:element ref="Születési_dátum" />
    <xs:element name="Email-cím" type="xs:string" />
  </xs:sequence>
  <xs:attribute name="felhasznalo_id" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="MegtekintiType">
  <xs:attribute name="film_id" type="xs:integer" use="required" />
  <xs:attribute name="felhasznalo_id" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="SzínészType">
  <xs:sequence>
    <xs:element name="Név" type="NévType">
    </xs:element>
    <xs:element ref="Születési_dátum" />
    <xs:element ref="Születési_hely" />
  </xs:sequence>
  <xs:attribute name="szinesz_id" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="SzerepelType">
  <xs:sequence>
    <xs:element name="Karakter_neve" type="xs:string" />
  </xs:sequence>
  <xs:attribute name="szerepel_id" type="xs:integer" use="required" />
  <xs:attribute name="film_id" type="xs:integer" use="required" />
  <xs:attribute name="szinesz_id" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="ÉlettársType">
  <xs:sequence>
    <xs:element name="Név" type="NévType"/>
    <xs:element ref="Születési_dátum" />
    <xs:element ref="Születési_hely" />
  </xs:sequence>
  <xs:attribute name="elettars_id" type="xs:integer" use="required" />
  <xs:attribute name="szinesz_id" type="xs:integer" use="required" />
</xs:complexType>

<xs:complexType name="DíjType">
  <xs:sequence>
    <xs:element name="Díj_típusa" type="xs:string" />
    <xs:element name="Feltételek" type="xs:string" />
    <xs:element name="Nyertes" type="xs:string" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="dij_id" type="xs:integer" use="required" />

```



```

</xs:complexType>

<xs:complexType name="ElnyeriType">
  <xs:attribute name="szinesz_id" type="xs:integer" use="required" />
  <xs:attribute name="dij_id" type="xs:integer" use="required" />
</xs:complexType>

<!-- Gyökérelem -->
<xs:element name="Filmatadbázis">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Film" type="FilmType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Értékelés" type="ÉrtékelésType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Kategória" type="KategóriaType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Tartozik" type="TartozikType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Felhasználó" type="FelhasználóType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Megtekinti" type="MegtekintiType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Színész" type="SzínészType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Szerepel" type="SzerepelType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Élettárs" type="ÉlettársType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Díj" type="DíjType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="Elnyeri" type="ElnyeriType" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- Elsődleges kulcsok-->
  <xs:key name="film_key">
    <xs:selector xpath="Film"></xs:selector>
    <xs:field xpath="@film_id"></xs:field>
  </xs:key>

  <xs:key name="ertekeles_key">
    <xs:selector xpath="Értékelés"></xs:selector>
    <xs:field xpath="@ertekeles_id"></xs:field>
  </xs:key>

  <xs:key name="kategoria_key">
    <xs:selector xpath="Kategória"></xs:selector>
    <xs:field xpath="@kategoria_id"></xs:field>
  </xs:key>

  <xs:key name="felhasznalo_key">
    <xs:selector xpath="Felhasználó"></xs:selector>
    <xs:field xpath="@felhasznalo_id"></xs:field>
  </xs:key>

  <xs:key name="szinesz_key">
    <xs:selector xpath="Színész"></xs:selector>
    <xs:field xpath="@szinesz_id"></xs:field>
  </xs:key>

```

```

<xs:key name="elettors_key">
  <xs:selector xpath="Élettárs"></xs:selector>
  <xs:field xpath="@elettors_id"></xs:field>
</xs:key>

<xs:key name="dij_key">
  <xs:selector xpath="Díj"></xs:selector>
  <xs:field xpath="@dij_id"></xs:field>
</xs:key>

<!-- Idegen kulcsok -->
<xs:keyref name="ertekeles_film_kulcs" refer="film_key">
  <xs:selector xpath="Értékelés" />
  <xs:field xpath="@film_id" />
</xs:keyref>
<xs:keyref name="tartozik_film_kulcs" refer="film_key">
  <xs:selector xpath="Tartozik" />
  <xs:field xpath="@film_id" />
</xs:keyref>
<xs:keyref name="tartozik_kategoria_kulcs" refer="kategoria_key">
  <xs:selector xpath="Tartozik" />
  <xs:field xpath="@kategoria_id" />
</xs:keyref>
<xs:keyref name="megtekinti_felhasznalo_kulcs" refer="felhasznalo_key">
  <xs:selector xpath="Megtekinti" />
  <xs:field xpath="@felhasznalo_id" />
</xs:keyref>
<xs:keyref name="szerepel_szinesz_kulcs" refer="szinesz_key">
  <xs:selector xpath="Szerepel" />
  <xs:field xpath="@szinesz_id" />
</xs:keyref>

<xs:keyref name="szerepel_film_kulcs" refer="film_key">
  <xs:selector xpath="Szerepel" />
  <xs:field xpath="@szinesz_id" />
</xs:keyref>

<xs:keyref name="elettors_szinesz_kulcs" refer="szinesz_key">
  <xs:selector xpath="Élettárs" />
  <xs:field xpath="@szinesz_id" />
</xs:keyref>

<xs:keyref name="elnyeri_dij_kulcs" refer="dij_key">
  <xs:selector xpath="Elnyeri" />
  <xs:field xpath="@dij_id" />
</xs:keyref>

</xs:element>

```

```
</xs:schema>
```

2. feladat

a) Adatolvasás

Az XML dokumentum beolvasása után felépítettem a dokumentum struktúráját egy string változóba úgy, hogy az elemeket egyenként beolvastam - az **appendChildNodes** függvény rekurzív hívásával - és hozzáfűztem őket.

A végén pedig a **saveXMLDocument** függvény elmenti a struktúrát egy új fájlba: „XMLReadGJ2N7R.xml” néven. Illetve a konzolra is kiíratásra kerül.

A program kódja:

```
package hu.domparse.gj2n7r;

import java.io.File;
import java.io.StringReader;

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 javax.xml.transform.stream.StreamSource;

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;

public class DomReadGJ2N7R {

    public static void main(String[] args){

        try{

            //XML dokumentum megnyitása
            File inputXML = new File("XML/XMLGJ2N7R.xml");
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dbBuilder = dbFactory.newDocumentBuilder();
            Document document = dbBuilder.parse(inputXML);

            //Kimeneti változó létrehozása
            String documentStructure = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n\n";

            //Gyökérelem beolvasása
            Node rootElement = document.getDocumentElement();
```

```

//Gyökérelem és attribútumainak csatolása
documentStructure += getRootAttributes(rootElement);

//Gyökérelem összes gyerek elemének beolvasása
NodeList rootChildren = rootElement.getChildNodes();

//Gyökérelem elemeinek csatolása a struktúrához
documentStructure += appendChildNodes(rootChildren);

//Dokumentum lezárása
documentStructure += "\n\n</" + rootElement.getNodeName() + ">\n";
System.out.println(documentStructure);

//Dokumentum mentése új XML fájlba
saveXMLDocument(documentStructure, "XMLReadGJ2N7R.xml");

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

}

private static void saveXMLDocument(String structure, String filePath) {
    try {
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();

        // StringReader létrehozása a struktúrának
        StringReader stringReader = new StringReader(structure);
        StreamSource source = new StreamSource(stringReader);

        // StreamResult létrehozása a kimeneti fájlnak
        StreamResult result = new StreamResult(new File(filePath));

        // Az XML string transformálása és mentése
        transformer.transform(source, result);

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

public static String appendChildNodes(NodeList children) {
    String structure = "";

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

        //Ha a csomó egy Element
        if (childNode.getNodeType() == Node.ELEMENT_NODE) {
            Element childElement = (Element) childNode;

```

```

//További gyerekelemek beolvasása
NodeList childNodes = childElement.getChildNodes();
boolean hasChildElements = false;
boolean hasText = false;

//Element gyerek csomópontjainak vizsgálása, van-e köztük Element
for(int j=0; j<childNodes.getLength(); j++){
    if(childNodes.item(j).getNodeType() == Node.ELEMENT_NODE){
        hasChildElements = true;
    }else if(childNodes.item(j).getNodeType() == Node.TEXT_NODE){
        hasText = true;
    }
}

//Ha az Elementnek vannak további gyerekelemei
if(hasChildElements){
    structure += "\n\t<" + childNode.getNodeName();

    //Element attribútumainak lekérdezése és a struktúrához csatolása
    NamedNodeMap attributes = childNode.getAttributes();
    for(int j=0; j<attributes.getLength(); j++){
        Node attribute = attributes.item(j);

        structure += " " + attribute;
    }

    structure += ">";

    structure += appendChildNodes(childNodes);
    structure += "\n\t</" + childNode.getNodeName() + ">\n";
} else if(hasText){ //Nincsenek további gyerekelemek; szöveg van
    structure += "\n\t<" + childElement.getNodeName() + ">" + childElement.getTextContent() + "</" +
childElement.getNodeName() + ">";
} else{ //Az az eset, ha nincs se szöveg, se további gyerekelem
    structure += "\n\t<" + childNode.getNodeName();
NamedNodeMap attributes = childNode.getAttributes();
    for(int j=0; j<attributes.getLength(); j++){
        Node attribute = attributes.item(j);

        structure += " " + attribute;
    }
    structure += ">" + "</" + childNode.getNodeName() + ">";
}

} else if (childNode.getNodeType() == Node.COMMENT_NODE) { //Komment hozzáadása
    structure += "\n\t<!--" + childNode.getTextContent() + "-->";
} else if (childNode.getNodeType() == Node.TEXT_NODE && !childNode.getTextContent().trim().isEmpty()) { //Üres
csomó hozzáadása új sorként
    structure += childNode.getTextContent() + "\n";
}

```

```

    }
}

return structure;
}
//Gyökérelem attribútumainak kiolvasása
public static String getRootAttributes(Node rootElement){

    String returnStructure = "<";

    NamedNodeMap rootElementAttributes = rootElement.getAttributes();
    returnStructure += rootElement.getNodeName();

    for(int i=0; i<rootElementAttributes.getLength(); i++){
        Node attribute = rootElementAttributes.item(i);
        returnStructure += " " + attribute;
    }

    returnStructure += ">\n";

    return returnStructure;
}
}

```

b) Adatmódosítás

Az XML dokumentum beolvasása után XPath segítségével lekérdeztem a módosítani kívánt adatokat a **modifyXMLFile** függvénnyel, a módosítás után pedig elmentettem a korábban is említett **saveXMLDocument** függvényemmel. A konzolra való kiíratáshoz Transformert használtam.

Az öt elvégzett módosítás a következő:

- Minden értékelés pontszámának növelése eggyel
- A díjak nyerteseihez 'Csuja Imre' hozzáadása
- 'Lara Worthington' nevű élettárs cseréje 'Sandra Bullock'-ra
- A második filmre mutató 'Szerepel' elem első egyedének, a színészre mutató idegen kulcs értékének megváltoztatása négyre
- A 'user3' nevű felhasználó születési dátumának megváltoztatása '2000-05-29'-re

A program kódja:

```
package hu.domparse.gj2n7r;

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

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 javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpression;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

import org.w3c.dom.DOMException;
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;

public class DomModifyGJ2N7R {
    public static void main(String[] args){
        try {

            String filePath = "XML/XMLGJ2N7RModify.xml";
            File inputFile = new File(filePath);
```



```

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document document = builder.parse(inputFile);
document.getDocumentElement().normalize();

//5 adat módosítása az XML fájlban
modifyXMLFile(document);

//XML fájl mentése
saveXMLDocument(document, filePath);

document.getDocumentElement().normalize();

}catch(Exception e){
    e.printStackTrace();
}
}
}
public static void modifyXMLFile(Document document) throws XPathExpressionException{

    try{
        XPathFactory xPathFactory = XPathFactory.newInstance();
        XPath xPath = xPathFactory.newXPath();

        //1. Minden értékelés pontszámának növelése egy ponttal
        System.out.println("Minden értékelés pontszámának növelése egy ponttal!");
        XPathExpression expression = xPath.compile("//Értékelés/Pontszám");
        NodeList Pontszámok = (NodeList) expression.evaluate(document, XPathConstants.NODESET);

        for(int i=0; i<Pontszámok.getLength(); i++){
            Node Pontszám = Pontszámok.item(i);

            //Előzetes pontszám lekérdezése és kiírása
            String pontszám = Pontszám.getTextContent().trim();

            System.out.println("A(z) " + (i+1) + ". jellegű pontszám: " + pontszám);

            //A pontszám konvertálása Double-re
            double currentValue = 0;
            try{
                currentValue = Double.parseDouble(pontszám);
            }catch(NumberFormatException e){
                e.printStackTrace();
            }

            //Pontszám növelése és visszakonvertálása Stringre
            Pontszám.setTextContent(String.valueOf(currentValue+1));

            System.out.println("A(z) " + (i+1) + ". módosítás utáni pontszám: " + Pontszám.getTextContent());
        }

        // 2. A díjak nyerteseihez egy-egy új Nyertes hozzáadása

```

```

System.out.println("Minden díj nyerteséhez Csuja Imre hozzáadása!");
XPathExpression expression2 = xPath.compile("//Díj");
NodeList awards = (NodeList) expression2.evaluate(document, XPathConstants.NODESET);

for(int i=0; i<awards.getLength(); i++){
    Element award = (Element) awards.item(i);

    //A díj Element meglévő struktúra kiírása kiírása
    System.out.println("A díj elem módosítás előtt:");
    consoleNodes(award);

    //Új nyertes létrehozása
    Element newWinner = document.createElement("Nyertes");

    newWinner.setTextContent("Csuja Imre");
    award.appendChild(newWinner);

    //Új díj Element új struktúra kiírása
    System.out.println("A díj elem módosítás után:");
    consoleNodes(award);

}

// 3. 'Lara Worthington' nevű élettárs csere 'Sandra Bullock'-ra
System.out.println("'Lara Worthington' nevű élettárs csere 'Sandra Bullock'-ra");
XPathExpression expression3 = xPath.compile("//Élettárs");
NodeList Partners = (NodeList) expression3.evaluate(document, XPathConstants.NODESET);

for(int i=0; i<Partners.getLength(); i++){
    Node partner = Partners.item(i);

    NodeList partnerChildren = partner.getChildNodes();
    for(int j=0; j<partnerChildren.getLength(); j++){
        Node partnerChild = partnerChildren.item(j);

        if(partnerChild.getNodeName().equals("Név")){
            NodeList nameChildren = partnerChild.getChildNodes();
            Node name = nameChildren.item(1);
            Node lastName = nameChildren.item(3);

            if(name.getTextContent().equals("Lara") && lastName.getTextContent().equals("Worthington")){
                //Meglévő élettárs nevének kiírása
                System.out.println("Korábbi élettárs: " + name.getTextContent() + " " + lastName.getTextContent());

                name.setTextContent("Sandra");
                lastName.setTextContent("Bullock");
            }
            //Új élettárs nevének kiírása
            System.out.println("Új élettárs: " + name.getTextContent() + " " + lastName.getTextContent());
        }
    }
}
}

```

```

    }

    // 4. A második filmre mutató 'Szerepel' Elementek első egyedének, a színészre mutató idegen kulcsának
    megváltoztatása a 4-esre
    System.out.println("A második filmre mutató 'Szerepel' Elementek első egyedének, a színészre mutató idegen
    kulcsának megváltoztatása a 4-esre");
    XPathExpression expression4 = XPath.compile("//Szerepel[@szerepel_id='1']");
    Node plays = (Node) expression4.evaluate(document, XPathConstants.NODE);

    System.out.println("A módosítás előtt:");
    consoleNodes(plays);

    NamedNodeMap playsAttributes = plays.getAttributes();
    for(int j=0; j<playsAttributes.getLength(); j++){
        Node attribute = playsAttributes.item(j);

        if(attribute.getNodeName().equals("film_id")){
            attribute.setNodeValue("4");
        }
    }

    System.out.println("A módosítás után:");
    consoleNodes(plays);
// 5. A 'user3' nevű felhasználó születési dátumának megváltoztatása
    System.out.println("A 'user3' nevű felhasználó születési dátumának megváltoztatása");
    XPathExpression expression5 = XPath.compile("//Felhasználó[Felhasználónév='user3']/Születési_dátum");
    Node userDateOfBirth = (Node) expression5.evaluate(document, XPathConstants.NODE);

    //A szülő Element meghatározása a kiíratáshoz
    Node user = userDateOfBirth.getParentNode();

    System.out.println("A módosítás előtt:");
    consoleNodes(user);

    //Születési dátum módosítása
    userDateOfBirth.setTextContent("2000-05-29");

    System.out.println("A módosítás után:");
    consoleNodes(user);

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

}

//Az Elementek kiíratása
public static void consoleNodes(Node node){
    String structure = "";
    structure += "<" + node.getNodeName() + " ";

```

```

NamedNodeMap nodeAttributes = node.getAttributes();
for(int i=0; i<nodeAttributes.getLength(); i++){
    Node attribute = nodeAttributes.item(i);

    structure += attribute;
}
structure += ">\n";

NodeList nodeChildren = node.getChildNodes();
for(int i=0; i<nodeChildren.getLength(); i++){
    Node child = nodeChildren.item(i);

    if(child.getNodeType() == Node.ELEMENT_NODE){
        structure += "\t<" + child.getNodeName() + ">" + child.getTextContent() + "</" + child.getNodeName() + ">\n";
    }
}

structure += "</" + node.getNodeName() + ">";

System.out.println(structure);

}

public static void saveXMLDocument(Document document, String filePath) {
    try {
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        DOMSource source = new DOMSource(document);
        StreamResult result = new StreamResult(new File(filePath));
        transformer.transform(source, result);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

c) Adatlekérdezés

Az XML dokumentum beolvasása után az öt lekérdezést öt különböző függvénnyel végeztem el. A konzolra való kiíratáshoz Transformert használtam.

Az öt lekérdezés a következő:

- Az első értékeléshez tartozó szövegek számának kiíratása
- Az 1900 előtt született felhasználók felhasználónevének kiíratása
- Amelyik filmhez több mint egy kategória tartozik, annak a filmnek a neve kerüljön kiíratásra
- A legjobb értékeléssel rendelkező film kerüljön kiíratásra
- A legtöbbszor kiosztott díj neve kerüljön kiíratásra

A program kódja:

```
package hu.domparse.gj2n7r;

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

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 javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpression;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

import org.w3c.dom.DOMException;
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;

public class DomQueryGJ2N7R {

    public static void main(String[] args){
        try{
            String filePath = "XML/XMLGJ2N7R.xml";

            File inputFile = new File(filePath);

            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse(inputFile);
```

```

document.getDocumentElement().normalize();

Element rootElement = document.getDocumentElement();
//5 lekérdezés elvégzése az 'XMLGJ2N7R.xml' fájlban
//1. Az első értékeléshez tartozó szövegek számának kiírása
ratingTextCounter(rootElement);

//2. Az 1900 előtt született felhasználók felhasználónevének kiírása
bornBefore1900(rootElement);

//3. Amelyik filmhez több mint egy kategória tartozik, annak a filmnek a neve kiírásra kerül
mostCategoryMovie(rootElement);

//4. A legjobb értékeléssel rendelkező film kerül kiírása
topRankedMovie(rootElement);

//5. A legtöbbször kiosztott díj neve kerül kiírásra
topWonAward(rootElement);

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

//1. Az első értékeléshez tartozó szövegek számának kiírása
public static void ratingTextCounter(Element rootElement){
    //Értékelések lekérdezése egy listába
    NodeList ratings = rootElement.getElementsByTagName("Értékelés");
    //Első Értékelés és a hozzá tartozó gyerekelemek lekérdezése
    Node firstRating = ratings.item(0);
    NodeList childNodes = firstRating.getChildNodes();
    int textCounter = 0;
    for(int i=0; i<childNodes.getLength(); i++){
        Node child = childNodes.item(i);

        if(child.getNodeName().equals("Értékelés_szövege")){
            textCounter++;
        }
    }

    //Az eredmény kiírása
    System.out.println("Az első 'Értékelés' elem " + textCounter + " darab szöveges értékelést tartalmaz");
}

//2. Az 1900 előtt született felhasználók felhasználónevének kiírása
public static void bornBefore1900(Element rootElement){
    //Felhasználók lekérdezése listába
    NodeList users = rootElement.getElementsByTagName("Felhasználó");

    //Felhasználók születési dátumának megvizsgálása egyenként
    for(int i=0; i<users.getLength(); i++){
        Node user = users.item(i);

```

```

//Felhasználó gyerekelemeinek lekérdezése
NodeList userChildren = user.getChildNodes();

//Feltételezve, hogy a felhasználó 1990 után született
boolean correctAge = false;

//Iterálás a felhasználó gyerekelemein
for(int j=0; j<userChildren.getLength(); j++){
    Node child = userChildren.item(j);

    //Ha a gyerek egy element
    if(child.getNodeType() == Node.ELEMENT_NODE){
        Element childElement = (Element) child;

        if(childElement.getNodeName().equals("Születési_dátum")){
            String[] dateOfBirth = childElement.getTextContent().split("-");
            if(Integer.parseInt(dateOfBirth[0]) < 1990){
                correctAge = true;
                j=0;
            }
        }else if (childElement.getNodeName().equals("Felhasználónév") && correctAge) {
            System.out.println("A " + childElement.getTextContent() + " nevű felhasználó 1990 előtt született");
            break;
        }
    }
}

}

}

}

//3. Amelyik filmhez több mint egy kategória tartozik, annak a filmnek a neve kiíratásra kerül
public static void mostCategoryMovie(Element rootElement){
    NodeList belong = rootElement.getElementsByTagName("Tartozik");

    //Kategória számláló tömb létrehozása
    int[] categoryCounter = new int[belong.getLength()];
    for (int i = 0; i < categoryCounter.length; i++) {
        categoryCounter[i] = 0;
    }

    //Iterálás a 'Tartozik' elemek között
    for(int i=0; i<belong.getLength(); i++){
        //Gyerekelemek azonosítása
        Node belongs = belong.item(i);

        NodeList belongChildren = belongs.getChildNodes();
        for(int j=0; j<belongChildren.getLength(); j++){
            Node belongChild = belongChildren.item(j);
            //Ha az elem egy gyerekelem
            if(belongChild.getNodeType() == Node.ELEMENT_NODE){
                //Kategóriák számának lekérdezése
                categoryCounter[i] = Integer.parseInt(belongChild.getTextContent());
            }
        }
    }
}

```

```

    }
}

//A legtöbb kategóriával rendelkező film keresése
int max = categoryCounter[0];
for (int i = 1; i < categoryCounter.length; i++) {
    if(categoryCounter[i] > max){
        max = categoryCounter[i];
    }
}

//Minden film kiírása, mely a maximális értékű kategóriaszámmal rendelkezik
for(int i=0; i<belong.getLength(); i++){
    //Ha rendelkezik a maximális értékű kategóriaszámmal
    if(categoryCounter[i] == max){
        Node belongs = belong.item(i);

        //A keresett film ID-je, azonosításra szorul
        int movieId = 0;

        //Gyerekelemek lekérdezése
        NamedNodeMap belongsAttributes = belongs.getAttributes();
        for(int j=0; j<belongsAttributes.getLength(); j++){
            Node attribute = belongsAttributes.item(j);
            if(attribute.getNodeName().equals("film_id")){ //Ha megvan a film_id attribútum, értékének mentése a movieId-
ba
                movieId = Integer.parseInt(attribute.getNodeValue());
            }
        }

        //Film megkeresése a filmId alapján
        NodeList filmek = rootElement.getElementsByTagName("Film");
        for(int j=0; j<filmek.getLength(); j++){
            Node film = filmek.item(j);

            //Film elem attribútumainak lekérdezése
            NamedNodeMap filmAttributes = film.getAttributes();
            for(int k=0; k<filmAttributes.getLength(); k++){
                Node attribute = filmAttributes.item(k);
                if(attribute.getNodeValue().equals(String.valueOf(movieId))){
                    //Meglett a keresett film; nevének kiírása
                    NodeList filmChildren = film.getChildNodes();

                    for(int l=0; l<filmChildren.getLength(); l++){
                        Node filmChild = filmChildren.item(l);

                        //Cím kiírása
                        if(filmChild.getNodeName().equals("Cím")){
                            System.out.println("A legtöbb kategóriával rendelkező film címe: " + filmChild.getTextContent());
                        }
                    }
                }
            }
        }
    }
}

```



```

    }
}

}
}
}

//4. A legjobb értékeléssel rendelkező film neve kerül kiírása
public static void topRankedMovie(Element rootElement){
    //Értékelések lekérdezése
    NodeList rankings = rootElement.getElementsByTagName("Értékelés");

    //A maximális pontszám deklarálása
    double maxRanking = 0;

    //Értékelések iterációja
    for(int i=0; i<rankings.getLength(); i++){
        Node ranking = rankings.item(i);

        //Gyerekelemek lekérdezése
        NodeList rankingChildren = ranking.getChildNodes();
        for(int j=0; j<rankingChildren.getLength(); j++){
            Node rankingChild = rankingChildren.item(j);

            //Ha a gyerekelem 'Pontszám', és értéke nagyobb mint az eddig talált legnagyobb pontszám
            if(rankingChild.getNodeName().equals("Pontszám") && Double.parseDouble(rankingChild.getTextContent()) >
maxRanking){
                maxRanking = Double.parseDouble(rankingChild.getTextContent());
            }
        }
    }

    String movieId = "";
    //Értékelések újra iterációja a legnagyobb pontszámmal rendelkező film megtalálásához
    for(int i=0; i<rankings.getLength(); i++){
        Node ranking = rankings.item(i);

        //Gyerekelemek lekérdezése
        NodeList rankingChildren = ranking.getChildNodes();
        for(int j=0; j<rankingChildren.getLength(); j++){
            Node rankingChild = rankingChildren.item(j);

            //Ha a gyerekelem 'Pontszám', és értéke megegyezik a legnagyobb pontszámmal
            if(rankingChild.getNodeName().equals("Pontszám") && Double.parseDouble(rankingChild.getTextContent()) ==
maxRanking){
                Node parent = rankingChild.getParentNode();

                //Szülő attribútumok lekérdezése
                NamedNodeMap parentAttributes = parent.getAttributes();
                for(int k=0; k<parentAttributes.getLength(); k++){
                    if(parentAttributes.item(k).getNodeName().equals("film_id")){
                        movieId = parentAttributes.item(k).getNodeValue();
                    }
                }
            }
        }
    }
}

```

```

    }
    }
    }
    }
}

//Film elemek lekérdezése
NodeList movies = rootElement.getElementsByTagName("Film");

for(int i=0; i<movies.getLength(); i++){
    Node movie = movies.item(i);

    //Film attribútumok lekérdezése
    NamedNodeMap movieAttributes = movie.getAttributes();
    for(int j=0; j<movieAttributes.getLength(); j++){
        //Ha az attribútum 'film_id', és értéke megegyezik a movieId-vel
        if(movieAttributes.item(j).getNodeName().equals("film_id")
movieAttributes.item(j).getNodeValue().equals(movieId)){
            //Film gyerekelemeinek lekérdezése
            NodeList movieChildren = movie.getChildNodes();

            for(int k=0; k<movieChildren.getLength(); k++){
                Node child = movieChildren.item(k);

                if(child.getNodeName().equals("Cím")){
                    System.out.println("A legjobb értékeléssel rendelkező film a " + child.getTextContent() + ", értékelése: " +
maxRanking);
                }
            }
        }
    }
}

}

//5. A legtöbbszor kiosztott díj neve kerül kiírásra
public static void topWonAward(Element rootElement){
    //Díj elemek lekérdezése
    NodeList awards = rootElement.getElementsByTagName("Díj");

    int[] winCounter = new int[awards.getLength()];
    for(int i=0; i<winCounter.length; i++){
        winCounter[i] = 0;
    }

    //Iterálás a díjakon
    for(int i=0; i<awards.getLength(); i++){
        Node award = awards.item(i);

        int counter = 0;
        //Iterálás a díj gyerekelemein
        for(int j=0; j<award.getChildNodes().getLength(); j++){

```

```

        Node awardChild = award.getChildNodes().item(j);
        if(awardChild.getNodeName().equals("Nyertes")){
            counter++;
        }
    }
    winCounter[i] = counter;
}

//Legtöbbször elnyert díjszám
int maxWinnerCount = winCounter[0];
for(int i=1; i<winCounter.length; i++){
    if(winCounter[i]>maxWinnerCount){
        maxWinnerCount = winCounter[i];
    }
}

//Újra iterálás a díjakon
for(int i=0; i<awards.getLength(); i++){
    Node award = awards.item(i);

    //Ha a díj elnyerésének száma a maximálissal megegyezik
    if(winCounter[i] == maxWinnerCount){
        for(int j=0; j<award.getChildNodes().getLength(); j++){
            //Gyerekelem lekérdezése
            Node awardChild = award.getChildNodes().item(j);

            if(awardChild.getNodeName().equals("Díj_típusa")){
                System.out.println("A legtöbbszőr kiosztott díj neve: " + awardChild.getTextContent());
            }
        }
    }
}
}
}
}
}
}
}
}

```

d) Adatírás

Ennél a feladatnál nem volt szükség dokumentum beolvasására, hiszen a feladat éppen ennek a dokumentumnak a létrehozása volt.

Ehhez egy univerzális feldolgozó függvényt írtam **createElement** névvel. A függvény a példányok minden adatát megkapja a paraméterlistában, ezt feldolgozza és hozzáfűzi az aadott elemet a dokumentumhoz.

Két kivétel volt, ahol nem használtam a createElement függvényt, mégpedig a **Megtekinti** és az **Elnyeri** kapcsolótáblák esetében. Ezekre manuálisan vittem fel az adatokat.

Ami még manuálisan történt, az a kommentelés. Minden új példány előtt egy '`<!--xy példányok -->`' kommentet hagytam.

Végül ezt is a **saveXMLDocument** függvénnyel mentettem, egy apró változtatással. A konzolra kiírás struktúráltságát a Properties osztály kiegészítésével valósítottam meg.

A program kódja:

```
package hu.domparse.gj2n7r;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
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.DOMException;
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 java.util.Properties;
import javax.xml.transform.OutputKeys;

public class DomWriteGJ2N7R {

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

        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dbBuilder = dbFactory.newDocumentBuilder();
        Document document = dbBuilder.newDocument();

        //Gyökérelem létrehozása
        Element rootElement = document.createElement("Filmadatbázis");
```

```

rootElement.setAttribute("xmlns:xs", "http://www.w3.org/2001/XMLSchema-instance");
rootElement.setAttribute("xs:noNamespaceSchemaLocation", "../xsd/XMLSchemaGJ2N7R.xsd");
document.appendChild(rootElement);

//Film komment hozzáadása
Node text = document.createTextNode("\n");
rootElement.appendChild(text);
text = document.createComment("Film példányok");
rootElement.appendChild(text);
//Film Elemek létrehozása
String[] movieIds = {"film_id"};
String[] movieElementNames = {"Cím", "Kiadás_éve", "Műfaj"};
String[][] movies = {
    {"Inception"}, {"The Shawshank Redemption"}, {"Avatar"}, {"The Dark Knight"}},
    {"2010"}, {"1994"}, {"2009"}, {"2008"}},
    {"Sci-fi"}, {"Dráma"}, {"Sci-fi"}, {"Akció"}
};
createElement(document, rootElement, "Film", movieIds, movies, movieElementNames);

//Értékelés komment hozzáadása
text = document.createTextNode("\n");
rootElement.appendChild(text);
text = document.createComment("Értékelés példányok");
rootElement.appendChild(text);

//Értékelés Elemek létrehozása
String[] rankIds = {"ertekeles_id", "film_id"};
String[] rankElementNames = {"Pontszám", "Értékelések_száma", "Értékelés_szöveg"};
String[][] ranks = {
    {"4.8"}, {"4.9"}, {"4.5"}, {"4.7"}},
    {"2000"}, {"1500"}, {"1200"}, {"1800"}},
    {
        "Nagyon jó film, érdemes megnézni!", "Lenyűgöző képi világ!", "Remek színészi alakítások!",
        "Minden idők egyik legjobb filmje!", "Nagyon izgalmas történet!",
        "Varázslatos filmélmény!", "Nagyszerű rendezés!", "Lenyűgöző látványvilág!",
        "Nagyon élvezetes film!", "Izgalmas cselekmény!"
    }
};
createElement(document, rootElement, "Értékelés", rankIds, ranks, rankElementNames);

//Kategória komment hozzáadása
text = document.createTextNode("\n");
rootElement.appendChild(text);
text = document.createComment("Kategória példányok");
rootElement.appendChild(text);
//Kategória Elemek létrehozása
String[] categoryIds = {"kategoria_id"};
String[] categoryElementNames = {"Kategória_név", "Leírás"};
String[][] categories = {
    {"Akció"}, {"Drama"}, {"Sci-fi"}},

```

```

        {"Izgalmas, pörgős jeleneteket tartalmazó filmek"}, {"Mély érzelmekre épülő filmek"}, {"Fantázia és tudományos
elemeket tartalmazó filmek"}}
    };
    createElement(document, rootElement, "Kategória", categoryIds, categories, categoryElementNames);

    //Tartozik komment hozzáadása
    text = document.createTextNode("\n");
    rootElement.appendChild(text);
    text = document.createComment("Tartozik kapcsolótábla példányok");
    rootElement.appendChild(text);

    // Tartozik kapcsolótábla Elemek létrehozása
    String[] belongsIds = {"film_id", "kategoria_id", "tartozik_id"};
    String[] belongsElementNames = {"Kategóriák_száma"};
    String[][] belongs = {
        {"2"}, {"1"}, {"3"}, {"4"}
    };
    createElement(document, rootElement, "Tartozik", belongsIds, belongs, belongsElementNames);

    //Felhasználó komment hozzáadása
    text = document.createTextNode("\n");
    rootElement.appendChild(text);
    text = document.createComment("Felhasználó példányok");
    rootElement.appendChild(text);

    // Felhasználó Elemek létrehozása
    String[] userIds = {"felhasznalo_id"};
    String[] userElementNames = {"Felhasználónév", "Születési_dátum", "Email-cím"};
    String[][] users = {
        {"user1"}, {"user2"}, {"user3"}, {"user4"},
        {"1985-05-15"}, {"1990-08-22"}, {"1988-03-10"}, {"1995-12-05"},
        {"user1@example.com"}, {"user2@example.com"}, {"user3@example.com"}, {"user4@example.com"}
    };
    createElement(document, rootElement, "Felhasználó", userIds, users, userElementNames);

    //Megtekinti komment hozzáadása
    text = document.createTextNode("\n");
    rootElement.appendChild(text);
    text = document.createComment("Megtekinti kapcsolótábla példányok");
    rootElement.appendChild(text);

    // Megtekinti kapcsolótábla Elemek létrehozása
    Element megtekinti1 = document.createElement("Megtekinti");
    megtekinti1.setAttribute("felhasznalo_id", "1");
    megtekinti1.setAttribute("film_id", "1");
    rootElement.appendChild(megtekinti1);

    Element megtekinti2 = document.createElement("Megtekinti");
    megtekinti2.setAttribute("felhasznalo_id", "2");
    megtekinti2.setAttribute("film_id", "2");
    rootElement.appendChild(megtekinti2);

```

```

Element megtekinti3 = document.createElement("Megtekinti");
megtekinti3.setAttribute("felhasznalo_id", "3");
megtekinti3.setAttribute("film_id", "3");
rootElement.appendChild(megtekinti3);

Element megtekinti4 = document.createElement("Megtekinti");
megtekinti4.setAttribute("felhasznalo_id", "4");
megtekinti4.setAttribute("film_id", "4");
rootElement.appendChild(megtekinti4);

//Színész komment hozzáadása
text = document.createTextNode("\n");
rootElement.appendChild(text);
text = document.createComment("Színész példányok");
rootElement.appendChild(text);

//Színész Elemek létrehozása
String[] actorIds = {"szinesz_id"};
String[] actorElementNames = {"Név", "Születési_dátum", "Születési_hely"};
String[][] actors = {
    {"Leonardo", "DiCaprio"}, {"Morgan", "Freeman"}, {"Sam", "Worthington"},
    {"1974-11-11"}, {"1937-06-01"}, {"1976-08-02"}},
    {"Los Angeles, Kalifornia"}, {"Memphis, Tennessee"}, {"Godalming, Egyesült Királyság"}}
};
createElement(document, rootElement, "Színész", actorIds, actors, actorElementNames);

//Szerepel komment hozzáadása
text = document.createTextNode("\n");
rootElement.appendChild(text);
text = document.createComment("Szerepel példányok");
rootElement.appendChild(text);
// Szerepel Elemek létrehozása
String[] actsIds = {"film_id", "szerepel_id", "szinesz_id"};
String[] actsElementNames = {"Karakter_neve"};
String[][] acts = {
    {"Dominic Cobb"}, {"Andy Dufresne"}, {"Jake Sully"}}
};
createElement(document, rootElement, "Szerepel", actsIds, acts, actsElementNames);

//Élettárs komment hozzáadása
text = document.createTextNode("\n");
rootElement.appendChild(text);
text = document.createComment("Élettárs példányok");
rootElement.appendChild(text);

// Élettárs Elemek létrehozása
String[] partnerIds = {"eletars_id", "szinesz_id"};
String[] partnerElementNames = {"Név", "Születési_dátum", "Születési_hely"};
String[][] partners = {
    {"Camila", "Morrone"}, {"Myrna", "Colley-Lee"}, {"Lara", "Washington"}},
    {"1997-06-16"}, {"1941-03-15"}, {"1976-08-02"}},
    {}
};

```

```

        {"Buenos Aires, Argentina"}, {"Milwaukee, Wisconsin"}, {"Godalming, Egyesült Királyság"}
    };
    createElement(document, rootElement, "Élettárs", partnerIds, partners, partnerElementNames);

    //Díj komment hozzáadása
    text = document.createTextNode("\n");
    rootElement.appendChild(text);
    text = document.createComment("Díj példányok");
    rootElement.appendChild(text);

    // Díj Elemek létrehozása
    String[] awardIds = {"dij_id"};
    String[] awardElementNames = {"Díj_típusa", "Feltételek", "Nyertes"};
    String[][] awards = {
        {"Oscar"}, {"Golden Globe"}, {"BAFTA"}},
        {"Legjobb film"}, {"Legjobb színész"}, {"Legjobb rendező"}},
        {
            {"Christopher Nolan", "Matthew McConaughey"},
            {"Leonardo DiCaprio", "Tom Hanks", "Emma Stone"},
            {"Alfonso Cuarón", "Greta Gerwig"}
        }
    };
    createElement(document, rootElement, "Díj", awardIds, awards, awardElementNames);
//Elnyeri komment hozzáadása
    text = document.createTextNode("\n");
    rootElement.appendChild(text);
    text = document.createComment("Elnyeri kapcsolótábla példányok");
    rootElement.appendChild(text);

    // Elnyeri kapcsolótábla Elemek létrehozása
    Element elnyeri1 = document.createElement("Elnyeri");
    elnyeri1.setAttribute("szinesz_id", "1");
    elnyeri1.setAttribute("dij_id", "1");
    rootElement.appendChild(elnyeri1);

    Element elnyeri2 = document.createElement("Elnyeri");
    elnyeri2.setAttribute("szinesz_id", "2");
    elnyeri2.setAttribute("dij_id", "2");
    rootElement.appendChild(elnyeri2);

    Element elnyeri3 = document.createElement("Elnyeri");
    elnyeri3.setAttribute("szinesz_id", "3");
    elnyeri3.setAttribute("dij_id", "3");
    rootElement.appendChild(elnyeri3);

    //XML fájl mentése
    document.getDocumentElement().normalize();
    saveXMLDocument(document, "XMLGJ2N7R1.xml");
}

public static void createElement(Document document, Element rootElement, String elementName, String[] ids, String[][]
elementValues, String[] elementNames){

```



```

//Dimenziók kiszámítása
int childElementCount = elementNames.length;
int elementCount = elementValues[0].length;

//Mennyi legyen ebből a bizonyos elemből
for(int i=0; i<elementCount; i++){
    //Új element létrehozása
    Element newElement = document.createElement(elementName);
    //Attribútumok hozzáfűzése
    for(int j=0; j<ids.length; j++){
        newElement.setAttribute(ids[j], String.valueOf(i+1));
    }

    //Gyerekelemek létrehozása
    for(int j=0; j<childElementCount; j++){
        //Hányszor fordulnak elő a gyerekelemek
        for(int k=0; k< elementValues[j][i].length; k++){
            Element newChildElement = document.createElement(elementNames[j]);
            newChildElement.setTextContent(elementValues[j][i][k]);

            newElement.appendChild(newChildElement);
        }
    }

    rootElement.appendChild(newElement);
}

}

public static void saveXMLDocument(Document document, String filePath) {
    try {
        //Transformer létrehozása
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();

        //Új 'Properties' objektum létrehozása a dokumentum struktúrálása érdekében
        Properties outputProperties = new Properties();
        outputProperties.setProperty(OutputKeys.INDENT, "yes"); //Indent property beállítása 'yes'-re
        outputProperties.setProperty("{http://xml.apache.org/xslt}indent-amount", "2"); //Szóköz beállítása 2-re
        transformer.setOutputProperties(outputProperties); //Maga a formátozás

        DOMSource source = new DOMSource(document);

        //Kiírás fájlba
        StreamResult result = new StreamResult(new File(filePath));
        transformer.transform(source, result);

        //Kiírás a konzolra
        StreamResult console = new StreamResult(System.out);
        transformer.transform(source, console);
    } catch (Exception e) {

```

```
e.printStackTrace();  
    }  
}  
}
```