JEGYZŐKÖNYV

Adatkezelés XML-ben

Féléves feladat

Képregény kiadó

Készítette: Györki Ádám
Neptunkód: A7TIJX
Dátum: 12.05

Feladat leírása:

Feladatom témáját foglalkozásom miatt a képregény piac, azon belül is a világ legnagyobb képregénypiaca, a japán. Ezt a képregénykiadók, a képregények (mangák) készítői és a könyvesboltok közötti kapcsolattal lehet megfelelő módon bemutatni. Mivel egy borzasztó nagy gépezetről van szó így egy, a feladathoz illő modellt lehet belőle szerkeszteni, azon keresztül pedig nagyon jól be lehet mutatni mi hogyan működik ezen a rendszeren belül. Mivel elemenként vizsgáljuk a szereplőket így a ránézésre bonyolult szisztémát részekre tudjuk bontani és ha így figyeljük meg az egészet akkor könnyen érthetővé válik a működés. Persze akár tovább is ki lehetne egészíteni ezt például a nyomdával is vagy más szereplőkkel, mint például a szerkesztők, vagy akár a teljes marketing gépezetet ide lehetne felsorakoztatni, ennél a modellénél viszont csak a leglényegesebb részeket vizsgáljuk, hogy hogyan jut el a képregény alkotótól az olvasóig a legújabb történet. Ahogy máskor is, most is fontosnak tartom, hogy autentikus legyen a feladat ezért valós adatokkal dolgoztam, ezeknek megfelelően utána néztem

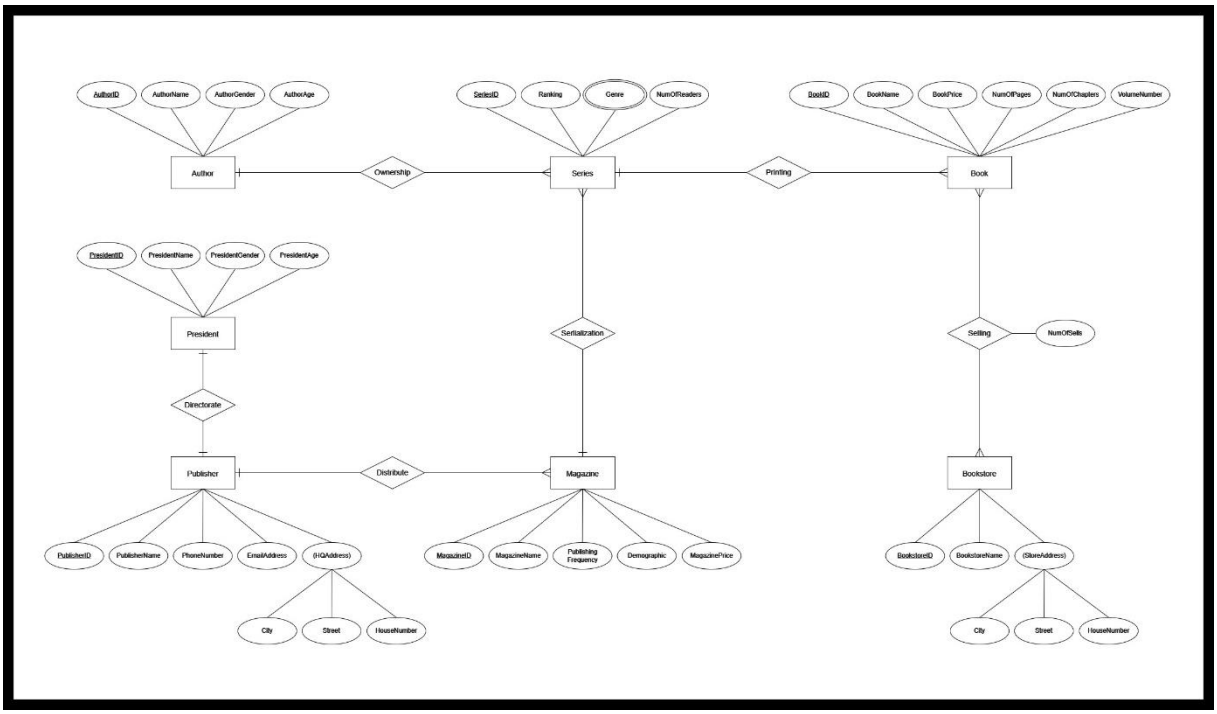
Maga a japán képregény piac a következőképpen működik: a képregénykészítők/mangakák elkészítik a fejezeteket, amiknek oldalszáma függ attól, hogy milyen megjelenési gyakoriságú magazinnál dolgoznak. Miután kész vannak ezt leadják és ez kikerül a magazinba, ami megjelenik a hét/hónap/negyed év elején. Ebben a magazinban mindig a legfrissebb fejezetek vannak. Ezeket tömeg gyártják és ezt adják el újságosoknál, könyvesboltokban vagy egyszerű vegyesboltokban. A magazin egy adott kiadóhoz tartozik, akik több magazint is futtatnak amelyek különböző megjelenési gyakorisággal rendelkeznek és más korosztálynak készülnek. A sounen tipikusan fiatal fiúknak,

a seinen felnőtt férfiaknak, a shojo fiatal lányoknak és a josei felnőtt nőknek készül. Minden kiadónak van egy elnöke aki főként hozza a döntéseket a kiadóval kapcsolatban. Miután összegyűlt egy adag (általában úgy 200-250 oldalnyi) fejezet, ezeket könyvekbe gyűjtik és így kiadják. Ezeket tudják venni az olvasók gyűjtésre és a szerző támogatására. Ezekben a magazinnal ellentétben csak egy szerző munkája van természetesen. Ezeket könyveket szórják ki különböző könyvesboltokban, ahol eladják őket a vevők számára.

1.feladat

1a) Az adatbázis ER modell tervezése

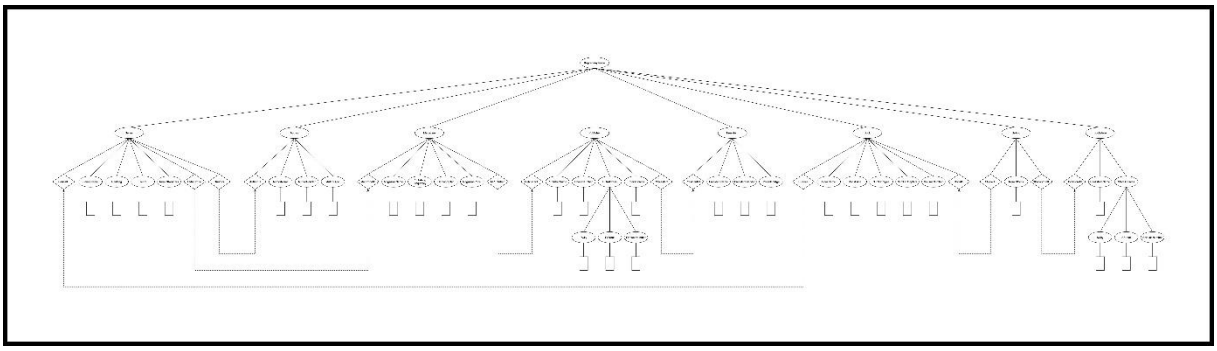
Az ER modell leírja a képregényiparban lévő szereplők közti kapcsolatokat, hogy hogyan viszonyulnak egymáshoz a felek. Ezeknek a feleknek vannak különféle adataik, amik fel vannak itt tüntetve valamint a kapcsolati tulajdonságuk is.
Az Author a képregényszerző, a Series a képregényszériák amiket az szerzők alkotnak és a magazinokban jelennek meg. A magazin (Magazine) a kiadó által vezetett nyomtatott képregények gyűjteménye ami adott időközönként jelenik meg. A kiadó (Publisher) amely üzemelteti a magazint, valamint ennek van egy igazgatója/vezetője. A könyvek (Book) azok a képregényszériák fejezeteiből összegyűjtött nyomtatott képregények amelyeket a könyvesboltokban (Bookstore) árulnak.



ERA7TIJX.jpg

1b) Az adatbázis konvertálása XDM modellre

Az XDM modellnél a legnagyobb hangsúlyt a a fa struktúra adja, erre kell lényegében átkonvertálnunk az eredeti relációs modellt. A különböző elemek alkotják a korábban említett szereplőket és itt is különbözőfél tulajdonságokkal látjuk el amik az elemek gyerekelemei lesznek. Ami még nehéz lehet ennél a folyamatnál azok az attribútumok és a közöttük fennálló kapcsolatok kialakítása. Eleve nem mindegy hogy milyen kapcsolat van az elemek között mert például ha több:több, akkor mindjárt be kell hoznunk egy újabb elemet (ez történt a Book és a Bookstore között. Ahogy az ER modellnél itt és érdemes figyelni a szépen kidolgozott küllemre és az átláthatóságra.



XDMA7TIJX.jpg

1c) Az XDM modell alapján XML dokumentum készítése

Ha az XDM modellt megfelelően elkészítettük akkor sok feladatunk nem is lesz az XML-el mivel lényegében ezzel megadtunk egy 1:1 sémát az XML kialakításához. A fa struktúrának megfelelően kialakítjuk az elemeket, hozzájuk írjuk a gyerekeiket és az attribútumokat. A többször előforduló elemeket próbálom apró változtatásokkal elnevezni, mint ahogy megkülönböztetem például az igazgató és az alkotó nemét vagy a kiadó és a könyvesbolt címét. Továbbra is valós adatokkal dolgozunk ezért nézhet ki furcsán elsőre a telefonszám, az email és a címek formátuma. A magazinokat yenben számoljuk azonban a könyveket dollárban, ez azért van így mivel a könyveket később lefordítják japánról angolra és különböző nyelvekre ami a világ minden pontjára elérést biztosít és a dollár egy jó számérték ha általánosságban akarjuk mérni világszerte az árát a könyvnek.

A Seriesben láthatjuk gyerekelemként a széria nevét, jelenlegi rankját a többihez képest, a műfaját és hogy mennyien olvassák. Attribútumoknál ki van emelve az ID-je valamint hogy melyik számú szerzőhöz és magazinhoz tartozik.

Az Authornál a szerző nevét, nemét és életkorát láthatjuk, valamint a saját IDját amire például hivatkozik a Series elem.
A Magazine-nál a magazin nevét látjuk, azt hogy milyen gyakran jelenik meg, hogy milyen korosztálynak szánták és hogy mennyibe is kerül egy átlag japán boltban. Ezen felül neki is van saját IDje és a tárol egy számot még ami megmutatja melyik kiadóhoz tartozik.
A Publishernél fel van tüntetve a kiadó neve, elérhetősége, mint email és telefon, valamint a címe, ami a városból, utcából és házszámból áll (ez az egész egy igen különös rendszert követ, amibe most nem mennék bele, a japán címek nagyon máshogy néznek ki mint itt Európában például, de a lényeg ugyan az. Ez is tartalmazza a saját IDjét és a kiadó vezetőjének, az elnöknek a számát.
A Bookban a könyvek általános adatait láthatjuk, mi a nevük, mi az áruk, hány fejezetből állnak, hány oldalból állnak és hányadik számú kötete a szériának a sorozatban.
A Bookstore a könyvesbolt nevét, és címét tartalmazza és utoljára marad a Selling ami összeköti a könyvet és a könyvesboltot ID tekintetében valamint tartalmaz egy eladási számot is a relációk fényében.

(XMLA7TIJX.xml) ---

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?><Kepregeny_kiado_A7TIJX>

    <!--A jelenleg futó képregényszériák-->

    <Series SAuthor="2" SMagazine="3" SeriesID="1">
        <SeriesName>Chainsaw Man</SeriesName>
        <Ranking>3#</Ranking>
        <Genre>
            <Subgenre>Action</Subgenre>
            <Subgenre>Supernatural</Subgenre>
            <Subgenre>Gore</Subgenre>
        </Genre>
        <NumOfReaders>750000</NumOfReaders>
    </Series>

    <Series SAuthor="1" SMagazine="2" SeriesID="2">
        <SeriesName>20th Century Boys</SeriesName>
        <Ranking>23#</Ranking>
        <Genre>
            <Subgenre>Drama</Subgenre>
            <Subgenre>Mystery</Subgenre>
            <Subgenre>Sci-fi</Subgenre>
        </Genre>
        <NumOfReaders>340000</NumOfReaders>
    </Series>

    <Series SAuthor="3" SMagazine="1" SeriesID="3">
        <SeriesName>Attack on Titan</SeriesName>
        <Ranking>2#</Ranking>
        <Genre>
            <Subgenre>Action</Subgenre>
            <Subgenre>Drama</Subgenre>
            <Subgenre>Suspense</Subgenre>
        </Genre>
        <NumOfReaders>2700000</NumOfReaders>
    </Series>

    <Series SAuthor="2" SMagazine="4" SeriesID="4">
        <SeriesName>Fire Punch</SeriesName>
        <Ranking>59#</Ranking>
        <Genre>
            <Subgenre>Action</Subgenre>
            <Subgenre>Supernatural</Subgenre>
            <Subgenre>Sci-fi</Subgenre>
        </Genre>
        <NumOfReaders>85000</NumOfReaders>
    </Series>
```

```xml
<!--A képregények szerzői-->

<Author AuthorID="1">
    <AuthorName>Urasawa Naoki</AuthorName>
    <AuthorGender>Male</AuthorGender>
    <AuthorAge>63</AuthorAge>
</Author>

<Author AuthorID="2">
    <AuthorName>Fujimoto Tatsuki</AuthorName>
    <AuthorGender>Male</AuthorGender>
    <AuthorAge>31</AuthorAge>
</Author>

<Author AuthorID="3">
    <AuthorName>Isayama Hajime</AuthorName>
    <AuthorGender>Male</AuthorGender>
    <AuthorAge>37</AuthorAge>
</Author>

<!--A magazinok amelyekben szereplenek a képregények-->

<Magazine MPublisher="3" MagazineID="1">
    <MagazineName>Monthly Shounen Magazine</MagazineName>
    <PublishingFrequency>Monthly</PublishingFrequency>
    <Demographic>Shounen</Demographic>
    <MagazinePrice>¥270</MagazinePrice>
</Magazine>

<Magazine MPublisher="2" MagazineID="2">
    <MagazineName>Big Comic Spirits</MagazineName>
    <PublishingFrequency>Weekly</PublishingFrequency>
    <Demographic>Seinen</Demographic>
    <MagazinePrice>¥280</MagazinePrice>
</Magazine>

<Magazine MPublisher="1" MagazineID="3">
    <MagazineName>Weekly Shounen Jump</MagazineName>
    <PublishingFrequency>Weekly</PublishingFrequency>
    <Demographic>Shounen</Demographic>
    <MagazinePrice>¥250</MagazinePrice>
</Magazine>

<Magazine MPublisher="1" MagazineID="4">
    <MagazineName>Shounen Jump+</MagazineName>
    <PublishingFrequency>Weekly</PublishingFrequency>
    <Demographic>Shounen</Demographic>
    <MagazinePrice>¥200</MagazinePrice>
</Magazine>
```

```
<!--A kiadók amelyek kiadják a magazinokat-->

<Publisher PPresident="2" PublisherID="1">
    <PublisherName>Shuiesha</PublisherName>
    <PhoneNumber>090-1731-1447</PhoneNumber>
    <EmailAddress>doe@shueisha.co.jp</EmailAddress>
    <HQAddress>
        <PCity>Tokyo</PCity>
        <PStreet>HitotsubashivChiyoda-ku</PStreet>
        <PHouseNumber>2-5-10</PHouseNumber>
    </HQAddress>
</Publisher>

<Publisher PPresident="3" PublisherID="2">
    <PublisherName>Shogakukan</PublisherName>
    <PhoneNumber>070-6386-1759</PhoneNumber>
    <EmailAddress>doe@shogakukan.co.jp</EmailAddress>
    <HQAddress>
        <PCity>Tokyo</PCity>
        <PStreet>Hitotsubashi Chiyoda-ku</PStreet>
        <PHouseNumber>2-3-1</PHouseNumber>
    </HQAddress>
</Publisher>

<Publisher PPresident="1" PublisherID="3">
    <PublisherName>Kodansha</PublisherName>
    <PhoneNumber>080-9598-3865</PhoneNumber>
    <EmailAddress>doe@kodansha.co.jp</EmailAddress>
    <HQAddress>
        <PCity>Tokyo</PCity>
        <PStreet>Otowa Bunkyo-ku </PStreet>
        <PHouseNumber>2-12-21</PHouseNumber>
    </HQAddress>
</Publisher>

<!--A kiadók elnökei-->

<President PresidentID="1">
    <PresidentName>Noma Yoshinobu</PresidentName>
    <PresidentGender>Male</PresidentGender>
    <PresidentAge>54</PresidentAge>
</President>

<President PresidentID="2">
    <PresidentName>Horiuchi Marue</PresidentName>
    <PresidentGender>Male</PresidentGender>
    <PresidentAge>72</PresidentAge>
</President>

<President PresidentID="3">
    <PresidentName>Oga Nobuhiro</PresidentName>
    <PresidentGender>Male</PresidentGender>
    <PresidentAge>75</PresidentAge>
</President>
```

```
<!--A képregényszériák nyomtatott könyv formátumban-->

<Book BSeries="1" BookID="1">
    <BookName>Chainsaw Man Vol.1</BookName>
    <BookPrice>$20</BookPrice>
    <NumOfPages>200</NumOfPages>
    <NumOfChapters>8</NumOfChapters>
    <VolumeNumber>1</VolumeNumber>
</Book>

<Book BSeries="1" BookID="2">
    <BookName>Chainsaw Man Vol.2</BookName>
    <BookPrice>$20</BookPrice>
    <NumOfPages>200</NumOfPages>
    <NumOfChapters>7</NumOfChapters>
    <VolumeNumber>2</VolumeNumber>
</Book>

<Book BSeries="3" BookID="3">
    <BookName>Attack on Titan Vol.1</BookName>
    <BookPrice>$25</BookPrice>
    <NumOfPages>230</NumOfPages>
    <NumOfChapters>7</NumOfChapters>
    <VolumeNumber>1</VolumeNumber>
</Book>

<Book BSeries="2" BookID="4">
    <BookName>20th Century Boys Vol.2</BookName>
    <BookPrice>$30</BookPrice>
    <NumOfPages>340</NumOfPages>
    <NumOfChapters>11</NumOfChapters>
    <VolumeNumber>2</VolumeNumber>
</Book>

<Book BSeries="4" BookID="5">
    <BookName>Fire Punch Vol.1</BookName>
    <BookPrice>$25</BookPrice>
    <NumOfPages>260</NumOfPages>
    <NumOfChapters>9</NumOfChapters>
    <VolumeNumber>1</VolumeNumber>
</Book>
```

```
<!--Eladások a könyvesboltokban-->

<Selling SBookID="1" SBookstoreID="1">
    <NumOfSells>4000</NumOfSells>
</Selling>

<Selling SBookID="1" SBookstoreID="2">
    <NumOfSells>3000</NumOfSells>
</Selling>

<Selling SBookID="1" SBookstoreID="3">
    <NumOfSells>7000</NumOfSells>
</Selling>

<Selling SBookID="2" SBookstoreID="1">
    <NumOfSells>3000</NumOfSells>
</Selling>

<Selling SBookID="2" SBookstoreID="2">
    <NumOfSells>2500</NumOfSells>
</Selling>

<Selling SBookID="2" SBookstoreID="3">
    <NumOfSells>4000</NumOfSells>
</Selling>

<Selling SBookID="3" SBookstoreID="1">
    <NumOfSells>8000</NumOfSells>
</Selling>

<Selling SBookID="3" SBookstoreID="3">
    <NumOfSells>10000</NumOfSells>
</Selling>

<Selling SBookID="4" SBookstoreID="2">
    <NumOfSells>3000</NumOfSells>
</Selling>

<Selling SBookID="5" SBookstoreID="3">
    <NumOfSells>2500</NumOfSells>
</Selling>
```

```
    <!--A könyvesboltok ahol eladjáka könyveket-->

    <Bookstore BookstoreID="1">
        <BookstoreName>Kinokuniya</BookstoreName>
        <StoreAddress>
            <SCity>Tokyo</SCity>
            <SStreet>3-chōme, Shinjuku</SStreet>
            <SHouseNumber>7-17</SHouseNumber>
        </StoreAddress>
    </Bookstore>

    <Bookstore BookstoreID="2">
        <BookstoreName>Daikanyama T-Site</BookstoreName>
        <StoreAddress>
            <SCity>Tokyo</SCity>
            <SStreet>Sarugaku-cho, Shibuya</SStreet>
            <SHouseNumber>17-5</SHouseNumber>
        </StoreAddress>
    </Bookstore>

    <Bookstore BookstoreID="3">
        <BookstoreName>Book First Umeda</BookstoreName>
        <StoreAddress>
            <SCity>Osaka</SCity>
            <SStreet>Kita-ku, Shibata</SStreet>
            <SHouseNumber>1-1-3</SHouseNumber>
        </StoreAddress>
    </Bookstore>

</Kepregeny_kiado_A7TIJX>
```

1d) Az XML dokumentum alapján XMLSchema készítése

Az XSD fájl az XML fájl létrehozásának segítségéül szolgál, előírja hogy mit, hogyan írhatunk bele a fájlba, akárcsak egy interface. Itt először egyszerű típusokat hoztam létre amelyek a legalapabb, legegyszerűbb elemeket képzik le, ezután ezeknek a referálásával komplex típusokat csináltam, majd saját típusokat. Ezeket végül szintén referálással felhasználom a végleges ábrázolásnál, ezért néz ki nagyon egyszerűen a felépítése a fájlnak. Fontos volt még hogy különböző korlátozásokat írjunk elő bizonyos elemekhez és az attribútumokhoz, például hogy mi legyen a típusa (int, string, saját), hogy kötelező szerepelnie e vagy hogy abból az adott elemből hány hozható létre az XML fájlban.

(XMLSchemaA7TIJX.xsd) ---

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

    <!--Képregény kiadó-->

    <xs:element name="Kepregeny_kiado_A7TIJX">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Series" type="SeriesType" minOccurs="1" maxOccurs="unbounded"/>
                <xs:element name="Author" type="AuthorType" minOccurs="1" maxOccurs="unbounded"/>
                <xs:element name="Magazine" type="MagazineTypes" minOccurs="1" maxOccurs="unbounded"/>
                <xs:element name="Publisher" type="PublisherType" minOccurs="1" maxOccurs="unbounded"/>
                <xs:element name="President" type="PresidentType" minOccurs="1" maxOccurs="unbounded"/>
                <xs:element name="Book" type="BookType" minOccurs="1" maxOccurs="unbounded"/>
                <xs:element name="Selling" type="SellingType" minOccurs="1" maxOccurs="unbounded"/>
                <xs:element name="Bookstore" type="BookstoreType" minOccurs="1" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <!--Főbb komplex típusok-->

    <xs:complexType name="SeriesType">
        <xs:sequence>
            <xs:element ref="SeriesName" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="Ranking" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="Genre" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="NumOfReaders" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="SeriesID" type="xs:integer" use="required"/>
        <xs:attribute name="SMagazine" type="xs:integer" use="required"/>
        <xs:attribute name="SAuthor" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:complexType name="AuthorType">
        <xs:sequence>
            <xs:element ref="AuthorName" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="AuthorGender" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="AuthorAge" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="AuthorID" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:complexType name="MagazineTypes">
        <xs:sequence>
            <xs:element ref="MagazineName" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="PublishingFrequency" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="Demographic" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="MagazinePrice" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="MagazineID" type="xs:integer" use="required"/>
        <xs:attribute name="MPublisher" type="xs:integer" use="required"/>
    </xs:complexType>
```

```
    <xs:complexType name="PublisherType">
        <xs:sequence>
            <xs:element ref="PublisherName" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="PhoneNumber" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="EmailAddress" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="HQAddress" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="PublisherID" type="xs:integer" use="required"/>
        <xs:attribute name="PPresident" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:complexType name="PresidentType">
        <xs:sequence>
            <xs:element ref="PresidentName" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="PresidentGender" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="PresidentAge" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="PresidentID" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:complexType name="BookType">
        <xs:sequence>
            <xs:element ref="BookName" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="BookPrice" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="NumOfPages" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="NumOfChapters" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="VolumeNumber" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="BookID" type="xs:integer" use="required"/>
        <xs:attribute name="BSeries" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:complexType name="SellingType">
        <xs:sequence>
            <xs:element name="NumOfSells" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="SBookID" type="xs:integer" use="required"/>
        <xs:attribute name="SBookstoreID" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:complexType name="BookstoreType">
        <xs:sequence>
            <xs:element ref="BookstoreName" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="StoreAddress" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="BookstoreID" type="xs:integer" use="required"/>
    </xs:complexType>
```

```
<!--Egyszerű típusok-->

    <!--Series-->
    <xs:element name="SeriesName" type="xs:string"/>
    <xs:element name="Ranking" type="RankingType"/>
    <xs:element name="NumOfReaders" type="xs:int"/>
    <xs:element name="Genre" type="GenreType"/>

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

    <!--Author-->
    <xs:element name="AuthorName" type="xs:string"/>
    <xs:element name="AuthorAge" type="xs:int"/>
    <xs:element name="AuthorGender" type="AuthorGenderType"/>

    <!--Magazine-->
    <xs:element name="MagazineName" type="xs:string"/>
    <xs:element name="PublishingFrequency" type="PublishingFrequencyType"/>
    <xs:element name="Demographic" type="DemographicType"/>
    <xs:element name="MagazinePrice" type="MagazinePriceType"/>

    <!--Publisher-->
    <xs:element name="PublisherName" type="xs:string"/>
    <xs:element name="PhoneNumber" type="PhoneNumberType"/>
    <xs:element name="EmailAddress" type="EmailAddressType"/>
    <xs:element name="HQAddress" type="HQAddressType"/>

    <xs:element name="PCity" type="xs:string"/>
    <xs:element name="PStreet" type="xs:string"/>
    <xs:element name="PHouseNumber" type="xs:string"/>

    <!--President-->
    <xs:element name="PresidentName" type="xs:string"/>
    <xs:element name="PresidentAge" type="xs:int"/>
    <xs:element name="PresidentGender" type="PresidentGenderType"/>

    <!--Book-->
    <xs:element name="BookName" type="xs:string"/>
    <xs:element name="BookPrice" type="BookPriceType"/>
    <xs:element name="NumOfPages" type="xs:int"/>
    <xs:element name="NumOfChapters" type="xs:int"/>
    <xs:element name="VolumeNumber" type="xs:int"/>

    <!--Selling-->
    <xs:element name="NumOfSells" type="xs:int"/>

    <!--Bookstore-->
    <xs:element name="BookstoreName" type="xs:string"/>
    <xs:element name="StoreAddress" type="StoreAddressType"/>

    <xs:element name="SCity" type="xs:string"/>
    <xs:element name="SStreet" type="xs:string"/>
    <xs:element name="SHouseNumber" type="xs:string"/>
```

```
    <!--Saját típusok-->

    <xs:simpleType name="RankingType">
        <xs:restriction base="xs:string">
            <xs:pattern value="([0-9])+#"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="AuthorGenderType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Male"/>
            <xs:enumeration value="Female"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="PublishingFrequencyType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Weekly"/>
            <xs:enumeration value="Bi-Weekly"/>
            <xs:enumeration value="Monthly"/>
            <xs:enumeration value="Quarterly"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="DemographicType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Shounen"/>
            <xs:enumeration value="Seinen"/>
            <xs:enumeration value="Shojo"/>
            <xs:enumeration value="Josei"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="MagazinePriceType">
        <xs:restriction base="xs:string">
            <xs:pattern value="¥([0-9])+"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="PhoneNumberType">
        <xs:restriction base="xs:string">
            <xs:pattern value="[0-9]{3}-[0-9]{4}-[0-9]{4}"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="EmailAddressType">
        <xs:restriction base="xs:string">
            <xs:pattern value="[^@]+@[^\.]+\..+"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="PresidentGenderType">
        <xs:restriction base="xs:string">
            <xs:pattern value="Male|Female"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="BookPriceType">
        <xs:restriction base="xs:string">
            <xs:pattern value="$([0-9])+"/>
        </xs:restriction>
    </xs:simpleType>
```

```
    <!--Komplex típusok-->

    <xs:complexType name="GenreType">
        <xs:sequence>
            <xs:element ref="Subgenre" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="HQAddressType">
        <xs:sequence>
            <xs:element ref="PCity" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="PStreet" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="PHouseNumber" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="StoreAddressType">
        <xs:sequence>
            <xs:element ref="SCity" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="SStreet" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="SHouseNumber" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

2. feladat

2a) adatolvasás

Az adatolvasónál létrehoztam az XML readert amivel betápláltam a programba az XML fájlt majd ezt Node-ok és NodeList-ek formájában feldaraboltam, majd ezeket eltároltam előre elkészített osztályok szerint az osztályokhoz tartozó metódusokkal. Azért maradtam a Listeknél mivel így jól átlátható így hogy kerültek tárolásra az adatok valamint a Listnek vannak hasznos funkciói amit később tudunk kamatoztatni. Az osztályoknál természetesen az adattagok az elemek és az attribútumok, az alattuk megtalálható metódusok pedig az előre létrehozott lista feltöltését szolgálják. Miután el lettek tárolva a Node-ok a listákba nagyon könnyen, egyszerű hivatkozásokkal ki tudjuk őket íratni konzolra. A feltöltő metódus úgy működik, hogy a NodeList-ből kiolvassuk a Nodeokat (az elemeket, gyerekelemeket) egy-egy ciklusban és a megfelelő tag-el rendelkező elemeket belerakjuk az újonnan létrehozott osztályokba. Egy következő metódus pedig azért felelős, hogy ezeket az elemeket List-be szedje.

(DomReadA7TIJX.java).xsd ---

```java
package hu.domparse.a7tijx;
import java.util.List;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.*;

import hu.domparse.a7tijx.elements.*;

import static hu.domparse.a7tijx.elements.Author.makeAuthorList;
import static hu.domparse.a7tijx.elements.Book.makeBookList;
import static hu.domparse.a7tijx.elements.Bookstore.makeBookstoreList;
import static hu.domparse.a7tijx.elements.Magazine.makeMagazineList;
import static hu.domparse.a7tijx.elements.President.makePresidentList;
import static hu.domparse.a7tijx.elements.Publisher.makePublisherList;
import static hu.domparse.a7tijx.elements.Selling.makeSellingList;
import static hu.domparse.a7tijx.elements.Series.makeSeriesList;

public class DomReadA7TIJX {
    public static void main(String[] args) {
        try{
            //XML dokumentum read
            DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = builderFactory.newDocumentBuilder();
            Document document = builder.parse("XMLA7TIJX.xml");
            document.getDocumentElement().normalize();

            //Node listák kiszedése a dokumentumból
            NodeList kkiado_a7tijx = document.getElementsByTagName("Kepregeny_kiado_A7TIJX");

            NodeList SeriesNodeList = document.getElementsByTagName("Series");
            NodeList AuthorNodeList = document.getElementsByTagName("Author");
            NodeList MagazineNodeList = document.getElementsByTagName("Magazine");
            NodeList PublisherNodeList = document.getElementsByTagName("Publisher");
            NodeList PresidentNodeList = document.getElementsByTagName("President");
            NodeList BookNodeList = document.getElementsByTagName("Book");
            NodeList SellingNodeList = document.getElementsByTagName("Selling");
            NodeList BookstoreNodeList = document.getElementsByTagName("Bookstore");

            //Node listákból elem listákat csinálunk
            List<Series> seriesList = makeSeriesList(SeriesNodeList);
            List<Author> authorList = makeAuthorList(AuthorNodeList);
            List<Magazine> magazineList = makeMagazineList(MagazineNodeList);
            List<Publisher> publisherList = makePublisherList(PublisherNodeList);
            List<President> presidentList = makePresidentList(PresidentNodeList);
            List<Book> bookList = makeBookList(BookNodeList);
            List<Selling> sellingList = makeSellingList(SellingNodeList);
            List<Bookstore> bookstoreList = makeBookstoreList(BookstoreNodeList);
```

```java
            //Teljes dokumentum kiírása egyedi strukturált formában listánként

            System.out.println("\n-------SERIES-------\n");

            for (Series s : seriesList) {
                System.out.println("Series:");
                System.out.println("Attributes: SeriesID=" + s.seriesID + ", SMagazine=" + s.sMagazine + ", SAuthor=" + s.sAuthor);
                System.out.print("Elements: SeriesName=" + s.seriesName + ", Ranking=" + s.ranking + ", NumOfReaders=" + s.numOfReaders + ", Genres: ");
                for (String subgenre : s.subGenre) {
                    System.out.print("Subgenre=" + subgenre + " ");
                }
                System.out.print("\n");
                System.out.println("-----------------------");
            }

            System.out.println("\n-------AUTHOR-------\n");

            for (Author a : authorList) {
                System.out.println("Author:");
                System.out.println("Attributes: AuthorID=" + a.authorID);
                System.out.println("Elements: AuthorName=" + a.authorName + ", AuthorGender=" + a.authorGender + ", AuthorAge=" + a.authorAge);
                System.out.println("-----------------------");
            }

            System.out.println("\n-------MAGAZINE-------\n");

            for (Magazine m : magazineList) {
                System.out.println("Magazine:");
                System.out.println("Attributes: MagazineID=" + m.magazineID + ", MPublisher=" + m.mPublisher);
                System.out.println("Elements: MagazineName=" + m.magazineName + ", PublishingFrequency=" + m.publishingFrequency + ", Demographic=" + m.demographic + ", MagazinePrice=" + m.magazinePrice);
                System.out.println("-----------------------");
            }

            System.out.println("\n-------PUBLISHER-------\n");

            for (Publisher pu : publisherList) {
                System.out.println("Publisher:");
                System.out.println("Attributes: PublisherID=" + pu.publisherID + ", PPresident=" + pu.pPresident);
                System.out.println("Elements: PublisherName=" + pu.publisherName + ", PhoneNumber=" + pu.phoneNumber + ", EmailAddress=" + pu.emailAddress + ", HQAddress: PCity=" + pu.pCity + ", PStreet=" + pu.pStreet + "
                System.out.println("-----------------------");
            }

            System.out.println("\n-------PRESIDENT-------\n");

            for (President pr : presidentList) {
                System.out.println("President:");
                System.out.println("Attributes: PresidentID=" + pr.presidentID);
                System.out.println("Elements: PresidentName=" + pr.presidentName + ", PresidentGender=" + pr.presidentGender + ", PresidentAge=" + pr.presidentAge);
                System.out.println("-----------------------");
            }
```



```java
        System.out.println("\n-------PRESIDENT-------\n");

        for (President pr : presidentList) {
            System.out.println("President:");
            System.out.println("Attributes: PresidentID=" + pr.presidentID);
            System.out.println("Elements: PresidentName=" + pr.presidentName + ", PresidentGender=" + pr.presidentGender + ", PresidentAge=" + pr.presidentAge);
            System.out.println("-----------------------");
        }

        System.out.println("\n-------BOOK-------\n");

        for (Book b : bookList) {
            System.out.println("Book:");
            System.out.println("Attributes: BookID=" + b.bookID + ", BSeries=" + b.bSeries);
            System.out.println("Elements: BookName=" + b.bookName + ", BookPrice=" + b.bookPrice + ", NumOfPages=" + b.numOfPages + ", NumOfChapters=" + b.numOfChapters + ", VolumeNumber=" + b.volumeNumber);
            System.out.println("-----------------------");
        }

        System.out.println("\n-------SELLING-------\n");

        for (Selling sl : sellingList) {
            System.out.println("Selling:");
            System.out.println("Attributes: SBookID=" + sl.sBookID + ", SBookstoreID=" + sl.sBookstoreID);
            System.out.println("Elements:  NumOfSells=" + sl.numOfSells);
            System.out.println("-----------------------");
        }

        System.out.println("\n-------BOOKSTORE-------\n");

        for (Bookstore bs : bookstoreList) {
            System.out.println("Bookstore:");
            System.out.println("Attributes: BookstoreID=" + bs.bookstoreID);
            System.out.println("Elements: BookstoreName=" + bs.bookstoreName + ", StoreAddress: SCity=" + bs.sCity + "SStreet=" + bs.sStreet + "SHouseNumber=" + bs.sHouseNumber);
            System.out.println("-----------------------");
        }
    }
    catch (Exception ee) {
    ee.printStackTrace();
    }
```

Az itt listákhoz felhasznált osztályok a következőket tartalmazzák:

```java
package hu.domparse.a7tijx.elements;

import org.w3c.dom.NodeList;

import java.util.ArrayList;
import java.util.List;

public class Author {
    //Attribútumok
    public int authorID;

    //Elemek
    public String authorName;
    public String authorGender;
    public int authorAge;

    //Author példány feltöltésére szolgál (segédfüggvény)
    public static Author authorLoad(NodeList authorNodes, int item){
        Author author = new Author();

        //Feltölti az elemeket
        for(int i = 0; i < authorNodes.item(item).getChildNodes().getLength(); i++){
            String nodeName = authorNodes.item(item).getChildNodes().item(i).getNodeName();
            String nodeContent = authorNodes.item(item).getChildNodes().item(i).getTextContent();
            if(nodeName.equals("AuthorName")){author.authorName = nodeContent;}
            else if(nodeName.equals("AuthorGender")){author.authorGender = nodeContent;}
            else if(nodeName.equals("AuthorAge")){author.authorAge = Integer.parseInt(nodeContent);}
        }

        //Feltölti az attribútumokat
        for(int i = 0; i < authorNodes.item(item).getAttributes().getLength(); i++){
            String nodeAttributeName = authorNodes.item(item).getAttributes().item(i).getNodeName();
            String nodeAttributeContent = authorNodes.item(item).getAttributes().item(i).getTextContent();

            if(nodeAttributeName.equals("AuthorID")){author.authorID = Integer.parseInt(nodeAttributeContent);}
        }

        return author;
    }

    //Készít egy Author osztály alapú listát az XML dokumentumból kapott Author Node listájából
    public static List<Author> makeAuthorList(NodeList authorNodeList){
        List<Author> authorList = new ArrayList<Author>();
        for(int i = 0; i < authorNodeList.getLength(); i++){
            authorList.add(authorLoad(authorNodeList, i));
        }
        return authorList;
    }
}

```

```java
package hu.domparse.a7tijx.elements;

import org.w3c.dom.NodeList;

import java.util.ArrayList;
import java.util.List;

public class Book {
    //Attribútumok
    public int bookID;
    public int bSeries;

    //Elemek
    public String bookName;
    public String bookPrice;
    public int numOfPages;
    public int numOfChapters;
    public int volumeNumber;


    //Book példány feltöltésére szolgál (segédfüggvény)
    public static Book bookLoad(NodeList bookNodes, int item){
        Book book = new Book();

        //Feltölti az elemeket
        for(int i = 0; i < bookNodes.item(item).getChildNodes().getLength(); i++){
            String nodeName = bookNodes.item(item).getChildNodes().item(i).getNodeName();
            String nodeContent = bookNodes.item(item).getChildNodes().item(i).getTextContent();
            if(nodeName.equals("BookName")){book.bookName = nodeContent;}
            else if(nodeName.equals("BookPrice")){book.bookPrice = nodeContent;}
            else if(nodeName.equals("NumOfPages")){book.numOfPages = Integer.parseInt(nodeContent);}
            else if(nodeName.equals("NumOfChapters")){book.numOfChapters = Integer.parseInt(nodeContent);}
            else if(nodeName.equals("VolumeNumber")){book.volumeNumber = Integer.parseInt(nodeContent);}
        }

        //Feltölti az attribútumokat
        for(int i = 0; i < bookNodes.item(item).getAttributes().getLength(); i++){
            String nodeAttributeName = bookNodes.item(item).getAttributes().item(i).getNodeName();
            String nodeAttributeContent = bookNodes.item(item).getAttributes().item(i).getTextContent();

            if(nodeAttributeName.equals("BookID")){book.bookID = Integer.parseInt(nodeAttributeContent);}
            else if(nodeAttributeName.equals("BSeries")){book.bSeries = Integer.parseInt(nodeAttributeContent);}
        }

        return book;
    }

    //Book egy Book osztály alapú listát az XML dokumentumból kapott Book Node listájából
    public static List<Book> makeBookList(NodeList bookNodeList){
        List<Book> bookList = new ArrayList<Book>();
        for(int i = 0; i < bookNodeList.getLength(); i++){
            bookList.add(bookLoad(bookNodeList, i));
        }
        return bookList;
    }
```

```java
package hu.domparse.a7tijx.elements;

import org.w3c.dom.NodeList;

import java.util.ArrayList;
import java.util.List;

public class Bookstore {
    //Attribútumok
    public int bookstoreID;

    //Elemek
    public String bookstoreName;
    public String sCity;
    public String sStreet;
    public String sHouseNumber;

    //Bookstore példány feltöltésére szolgál (segédfüggvény)
    public static Bookstore bookstoreLoad(NodeList bookstoreNodes, int item){
        Bookstore bookstore = new Bookstore();

        //Feltölti az elemeket
        for(int i = 0; i < bookstoreNodes.item(item).getChildNodes().getLength(); i++){
            String nodeName = bookstoreNodes.item(item).getChildNodes().item(i).getNodeName();
            String nodeContent = bookstoreNodes.item(item).getChildNodes().item(i).getTextContent();
            if(nodeName.equals("BookstoreName")){bookstore.bookstoreName = nodeContent;}
            else if(nodeName.equals("StoreAddress")){
                for(int j = 0; j < bookstoreNodes.item(item).getChildNodes().item(i).getChildNodes().getLength(); j++){
                    String addressName = bookstoreNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getNodeName();
                    String addressContent = bookstoreNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getTextContent();
                    if(addressName.equals("SCity")){bookstore.sCity = addressContent;}
                    else if(addressName.equals("SStreet")){bookstore.sStreet = addressContent;}
                    else if(addressName.equals("SHouseNumber")){bookstore.sHouseNumber = addressContent;}
                }
            }
        }

        //Feltölti az attribútumokat
        for(int i = 0; i < bookstoreNodes.item(item).getAttributes().getLength(); i++){
            String nodeAttributeName = bookstoreNodes.item(item).getAttributes().item(i).getNodeName();
            String nodeAttributeContent = bookstoreNodes.item(item).getAttributes().item(i).getTextContent();

            if(nodeAttributeName.equals("BookstoreID")){bookstore.bookstoreID = Integer.parseInt(nodeAttributeContent);}
        }

        return bookstore;
    }

    //Készít egy Bookstore osztály alapú listát az XML dokumentumból kapott Bookstore Node listájából
    public static List<Bookstore> makeBookstoreList(NodeList bookstoreNodeList){
        List<Bookstore> bookstoreList = new ArrayList<Bookstore>();
        for(int i = 0; i < bookstoreNodeList.getLength(); i++){
            bookstoreList.add(bookstoreLoad(bookstoreNodeList, i));
        }
        return bookstoreList;
```

```java
package hu.domparse.a7tijx.elements;

import org.w3c.dom.NodeList;

import java.util.ArrayList;
import java.util.List;

public class Magazine {
    //Attribútumok
    public int magazineID;
    public int mPublisher;

    //Elemek
    public String magazineName;
    public String publishingFrequency;
    public String demographic;
    public String magazinePrice;


    //Magazine példány feltöltésére szolgál (segédfüggvény)
    public static Magazine magazineLoad(NodeList magazineNodes, int item){
        Magazine magazine = new Magazine();

        //Feltölti az elemeket
        for(int i = 0; i < magazineNodes.item(item).getChildNodes().getLength(); i++){
            String nodeName = magazineNodes.item(item).getChildNodes().item(i).getNodeName();
            String nodeContent = magazineNodes.item(item).getChildNodes().item(i).getTextContent();
            if(nodeName.equals("MagazineName")){magazine.magazineName = nodeContent;}
            else if(nodeName.equals("PublishingFrequency")){magazine.publishingFrequency = nodeContent;}
            else if(nodeName.equals("Demographic")){magazine.demographic = nodeContent;}
            else if(nodeName.equals("MagazinePrice")){magazine.magazinePrice = nodeContent;}
        }

        //Feltölti az attribútumokat
        for(int i = 0; i < magazineNodes.item(item).getAttributes().getLength(); i++){
            String nodeAttributeName = magazineNodes.item(item).getAttributes().item(i).getNodeName();
            String nodeAttributeContent = magazineNodes.item(item).getAttributes().item(i).getTextContent();

            if(nodeAttributeName.equals("MagazineID")){magazine.magazineID = Integer.parseInt(nodeAttributeContent);}
            else if(nodeAttributeName.equals("MPublisher")){magazine.mPublisher = Integer.parseInt(nodeAttributeContent);}
        }

        return magazine;
    }

    //Készít egy Magazine osztály alapú listát az XML dokumentumból kapott Magazine Node listájából
    public static List<Magazine> makeMagazineList(NodeList magazineNodeList){
        List<Magazine> magazineList = new ArrayList<Magazine>();
        for(int i = 0; i < magazineNodeList.getLength(); i++){
            magazineList.add(magazineLoad(magazineNodeList, i));
        }
        return magazineList;
    }
}

```

```java
package hu.domparse.a7tijx.elements;

import org.w3c.dom.NodeList;

import java.util.ArrayList;
import java.util.List;

public class President {
    //Attribútumok
    public int presidentID;

    //Elemek
    public String presidentName;
    public String presidentGender;
    public int presidentAge;

    //President példány feltöltésére szolgál (segédfüggvény)
    public static President presidentLoad(NodeList presidentNodes, int item){
        President president = new President();

        //Feltölti az elemeket
        for(int i = 0; i < presidentNodes.item(item).getChildNodes().getLength(); i++){
            String nodeName = presidentNodes.item(item).getChildNodes().item(i).getNodeName();
            String nodeContent = presidentNodes.item(item).getChildNodes().item(i).getTextContent();
            if(nodeName.equals("PresidentName")){president.presidentName = nodeContent;}
            else if(nodeName.equals("PresidentGender")){president.presidentGender = nodeContent;}
            else if(nodeName.equals("PresidentAge")){president.presidentAge = Integer.parseInt(nodeContent);}
        }

        //Feltölti az attribútumokat
        for(int i = 0; i < presidentNodes.item(item).getAttributes().getLength(); i++){
            String nodeAttributeName = presidentNodes.item(item).getAttributes().item(i).getNodeName();
            String nodeAttributeContent = presidentNodes.item(item).getAttributes().item(i).getTextContent();

            if(nodeAttributeName.equals("PresidentID")){president.presidentID = Integer.parseInt(nodeAttributeContent);}
        }

        return president;
    }

    //Készít egy President osztály alapú listát az XML dokumentumból kapott President Node listájából
    public static List<President> makePresidentList(NodeList presidentNodeList){
        List<President> presidentList = new ArrayList<President>();
        for(int i = 0; i < presidentNodeList.getLength(); i++){
            presidentList.add(presidentLoad(presidentNodeList, i));
        }
        return presidentList;
    }
}

```

```java
package hu.domparse.a7tijx.elements;

import org.w3c.dom.NodeList;

import java.util.ArrayList;
import java.util.List;

public class Publisher {
    //Attribútumok
    public int publisherID;
    public int pPresident;

    //Elemek
    public String publisherName;
    public String phoneNumber;
    public String emailAddress;
    public String pCity;
    public String pStreet;
    public String pHouseNumber;

    //Publisher példány feltöltésére szolgál (segédfüggvény)
    public static Publisher publisherLoad(NodeList publisherNodes, int item){
        Publisher publisher = new Publisher();

        //Feltölti az elemeket
        for(int i = 0; i < publisherNodes.item(item).getChildNodes().getLength(); i++){
            String nodeName = publisherNodes.item(item).getChildNodes().item(i).getNodeName();
            String nodeContent = publisherNodes.item(item).getChildNodes().item(i).getTextContent();
            if(nodeName.equals("PublisherName")){publisher.publisherName = nodeContent;}
            else if(nodeName.equals("PhoneNumber")){publisher.phoneNumber = nodeContent;}
            else if(nodeName.equals("EmailAddress")){publisher.emailAddress = nodeContent;}
            else if(nodeName.equals("HQAddress")){
                for(int j = 0; j < publisherNodes.item(item).getChildNodes().item(i).getChildNodes().getLength(); j++){
                    String addressName = publisherNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getNodeName();
                    String addressContent = publisherNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getTextContent();
                    if(addressName.equals("PCity")){publisher.pCity = addressContent;}
                    else if(addressName.equals("PStreet")){publisher.pStreet = addressContent;}
                    else if(addressName.equals("PHouseNumber")){publisher.pHouseNumber = addressContent;}
                }
            }
        }

        //Feltölti az attribútumokat
        for(int i = 0; i < publisherNodes.item(item).getAttributes().getLength(); i++){
            String nodeAttributeName = publisherNodes.item(item).getAttributes().item(i).getNodeName();
            String nodeAttributeContent = publisherNodes.item(item).getAttributes().item(i).getTextContent();

            if(nodeAttributeName.equals("PublisherID")){publisher.publisherID = Integer.parseInt(nodeAttributeContent);}
            else if(nodeAttributeName.equals("PPresident")){publisher.pPresident = Integer.parseInt(nodeAttributeContent);}
        }

        return publisher;
    }

    //Készít egy Publisher osztály alapú listát az XML dokumentumból kapott Publisher Node listájából
    public static List<Publisher> makePublisherList(NodeList publisherNodeList){
        List<Publisher> publisherList = new ArrayList<Publisher>();
        for(int i = 0; i < publisherNodeList.getLength(); i++){
            publisherList.add(publisherLoad(publisherNodeList, i));
        }
        return publisherList;
    }
}

```

DOMParseA7tijx > DomParseA7tijx > hu > domparse > a7tijx > elements > Selling.java

```java
package hu.domparse.a7tijx.elements;

import org.w3c.dom.NodeList;

import java.util.ArrayList;
import java.util.List;

public class Selling {
    //Attribútumok
    public int sBookID;
    public int sBookstoreID;

    //Elemek
    public int numOfSells;

    //Selling példány feltöltésére szolgál (segédfüggvény)
    public static Selling sellingLoad(NodeList sellingNodes, int item){
        Selling selling = new Selling();

        //Feltölti az elemeket
        for(int i = 0; i < sellingNodes.item(item).getChildNodes().getLength(); i++){
            String nodeName = sellingNodes.item(item).getChildNodes().item(i).getNodeName();
            String nodeContent = sellingNodes.item(item).getChildNodes().item(i).getTextContent();

            if(nodeName.equals("NumOfSells")){selling.numOfSells = Integer.parseInt(nodeContent);}
        }

        //Feltölti az attribútumokat
        for(int i = 0; i < sellingNodes.item(item).getAttributes().getLength(); i++){
            String nodeAttributeName = sellingNodes.item(item).getAttributes().item(i).getNodeName();
            String nodeAttributeContent = sellingNodes.item(item).getAttributes().item(i).getTextContent();

            if(nodeAttributeName.equals("SBookID")){selling.sBookID = Integer.parseInt(nodeAttributeContent);}
            else if(nodeAttributeName.equals("SBookstoreID")){selling.sBookstoreID = Integer.parseInt(nodeAttributeContent);}
        }

        return selling;
    }

    //Készít egy Selling osztály alapú listát az XML dokumentumból kapott Selling Node listájából
    public static List<Selling> makeSellingList(NodeList sellingNodeList){
        List<Selling> sellingList = new ArrayList<Selling>();
        for(int i = 0; i < sellingNodeList.getLength(); i++){
            sellingList.add(sellingLoad(sellingNodeList, i));
        }
        return sellingList;
    }
}

```

```java
package hu.domparse.a7tijx.elements;

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

import java.util.ArrayList;
import java.util.List;

public class Series{
    //Attribútumok
    public int seriesID;
    public int sMagazine;
    public int sAuthor;

    //Elemek
    public String seriesName;
    public String ranking;
    public int numOfReaders;
    public List<String> subGenre;

    //Series példány feltöltésére szolgál (segédfüggvény)
    public static Series seriesLoad(NodeList seriesNodes, int item){
        Series series = new Series();

        //Feltölti az elemeket
        for(int i = 0; i < seriesNodes.item(item).getChildNodes().getLength(); i++){
            String nodeName = seriesNodes.item(item).getChildNodes().item(i).getNodeName();
            String nodeContent = seriesNodes.item(item).getChildNodes().item(i).getTextContent();
            if(nodeName.equals("SeriesName")){series.seriesName = nodeContent;}
            else if(nodeName.equals("Ranking")){series.ranking = nodeContent;}
            else if(nodeName.equals("NumOfReaders")){series.numOfReaders = Integer.parseInt(nodeContent);}
            else if(nodeName.equals("Genre")){
                List<String> subgenreList = new ArrayList<String>();
                for (int j = 0; j < seriesNodes.item(item).getChildNodes().item(i).getChildNodes().getLength(); j++ ) {
                    if(seriesNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getNodeType() == Node.ELEMENT_NODE){
                        subgenreList.add(seriesNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getTextContent());
                    }

                }
                series.subGenre = subgenreList;
            }
        }

        //Feltölti az attribútumokat
        for(int i = 0; i < seriesNodes.item(item).getAttributes().getLength(); i++){
            String nodeAttributeName = seriesNodes.item(item).getAttributes().item(i).getNodeName();
            String nodeAttributeContent = seriesNodes.item(item).getAttributes().item(i).getTextContent();

            if(nodeAttributeName.equals("SeriesID")){series.seriesID = Integer.parseInt(nodeAttributeContent);}
            else if(nodeAttributeName.equals("SMagazine")){series.sMagazine = Integer.parseInt(nodeAttributeContent);}
            else if(nodeAttributeName.equals("SAuthor")){series.sAuthor = Integer.parseInt(nodeAttributeContent);}
        }

        return series;
    }

```

Ezt kapjuk kiírásként a konzolon:

```
-------SERIES-------

Series:
Attributes: SeriesID=1, SMagazine=3, SAuthor=2
Elements: SeriesName=Chainsaw Man, Ranking=3#, NumOfReaders=750000, Genres: Subgenre=Action Subgenre=Supernatural Subgenre=Gore
----------------------
Series:
Attributes: SeriesID=2, SMagazine=2, SAuthor=1
Elements: SeriesName=20th Century Boys, Ranking=23#, NumOfReaders=340000, Genres: Subgenre=Drama Subgenre=Mystery Subgenre=Sci-fi
----------------------
Series:
Attributes: SeriesID=3, SMagazine=1, SAuthor=3
Elements: SeriesName=Attack on Titan, Ranking=2#, NumOfReaders=2700000, Genres: Subgenre=Action Subgenre=Drama Subgenre=Suspense
----------------------
Series:
Attributes: SeriesID=4, SMagazine=4, SAuthor=2
Elements: SeriesName=Fire Punch, Ranking=59#, NumOfReaders=85000, Genres: Subgenre=Action Subgenre=Supernatural Subgenre=Sci-fi
----------------------
```

```
-------AUTHOR-------

Author:
Attributes: AuthorID=1
Elements: AuthorName=Urasawa Naoki, AuthorGender=Male, AuthorAge=63
-----------------------
Author:
Attributes: AuthorID=2
Elements: AuthorName=Fujimoto Tatsuki, AuthorGender=Male, AuthorAge=31
-----------------------
Author:
Attributes: AuthorID=3
Elements: AuthorName=Isayama Hajime, AuthorGender=Male, AuthorAge=37
-----------------------
```

```
-------MAGAZINE-------

Magazine:
Attributes: MagazineID=1, MPublisher=3
Elements: MagazineName=Monthly Shounen Magazine, PublishingFrequency=Monthly, Demographic=Shounen, MagazinePrice=¥270
----------------------
Magazine:
Attributes: MagazineID=2, MPublisher=2
Elements: MagazineName=Big Comic Spirits, PublishingFrequency=Weekly, Demographic=Seinen, MagazinePrice=¥280
----------------------
Magazine:
Attributes: MagazineID=3, MPublisher=1
Elements: MagazineName=Weekly Shounen Jump, PublishingFrequency=Weekly, Demographic=Shounen, MagazinePrice=¥250
----------------------
Magazine:
Attributes: MagazineID=4, MPublisher=1
Elements: MagazineName=Shounen Jump+, PublishingFrequency=Weekly, Demographic=Shounen, MagazinePrice=¥200
----------------------
```

```
-------PUBLISHER-------

Publisher:
Attributes: PublisherID=1, PPresident=2
Elements: PublisherName=Shuiesha, PhoneNumber=090-1731-1447, EmailAddress=doe@shueisha.co.jp, HQAddress: PCity=Tokyo, PStreet=HitotsubashivChiyoda-ku, PHouseNumber=2-5-10
----------------------
Publisher:
Attributes: PublisherID=2, PPresident=3
Elements: PublisherName=Shogakukan, PhoneNumber=070-6386-1759, EmailAddress=doe@shogakukan.co.jp, HQAddress: PCity=Tokyo, PStreet=Hitotsubashi Chiyoda-ku, PHouseNumber=2-3-1
----------------------
Publisher:
Attributes: PublisherID=3, PPresident=1
Elements: PublisherName=Kodansha, PhoneNumber=080-9598-3865, EmailAddress=doe@kodansha.co.jp, HQAddress: PCity=Tokyo, PStreet=Otowa Bunkyo-ku , PHouseNumber=2-12-21
----------------------
```

```
-------PRESIDENT-------

President:
Attributes: PresidentID=1
Elements: PresidentName=Noma Yoshinobu, PresidentGender=Male, PresidentAge=54
----------------------
President:
Attributes: PresidentID=2
Elements: PresidentName=Horiuchi Marue, PresidentGender=Male, PresidentAge=72
----------------------
President:
Attributes: PresidentID=3
Elements: PresidentName=Oga Nobuhiro, PresidentGender=Male, PresidentAge=75
----------------------
```

```
-------BOOK-------

Book:
Attributes: BookID=1, BSeries=1
Elements: BookName=Chainsaw Man Vol.1, BookPrice=$20, NumOfPages=200, NumOfChapters=8, VolumeNumber=1
----------------------
Book:
Attributes: BookID=2, BSeries=1
Elements: BookName=Chainsaw Man Vol.2, BookPrice=$20, NumOfPages=200, NumOfChapters=7, VolumeNumber=2
----------------------
Book:
Attributes: BookID=3, BSeries=3
Elements: BookName=Attack on Titan Vol.1, BookPrice=$25, NumOfPages=230, NumOfChapters=7, VolumeNumber=1
----------------------
Book:
Attributes: BookID=4, BSeries=2
Elements: BookName=20th Century Boys Vol.2, BookPrice=$30, NumOfPages=340, NumOfChapters=11, VolumeNumber=2
----------------------
Book:
Attributes: BookID=5, BSeries=4
Elements: BookName=Fire Punch Vol.1, BookPrice=$25, NumOfPages=260, NumOfChapters=9, VolumeNumber=1
----------------------
```

```
-------SELLING-------

Selling:
Attributes: SBookID=1, SBookstoreID=1
Elements:  NumOfSells=4000
----------------------
Selling:
Attributes: SBookID=1, SBookstoreID=2
Elements:  NumOfSells=3000
----------------------
Selling:
Attributes: SBookID=1, SBookstoreID=3
Elements:  NumOfSells=7000
----------------------
Selling:
Attributes: SBookID=2, SBookstoreID=1
Elements:  NumOfSells=3000
----------------------
Selling:
Attributes: SBookID=2, SBookstoreID=2
Elements:  NumOfSells=2500
----------------------
```

```
Selling:
Attributes: SBookID=2, SBookstoreID=3
Elements:  NumOfSells=4000
----------------------
Selling:
Attributes: SBookID=3, SBookstoreID=1
Elements:  NumOfSells=8000
----------------------
Selling:
Attributes: SBookID=3, SBookstoreID=3
Elements:  NumOfSells=10000
----------------------
Selling:
Attributes: SBookID=4, SBookstoreID=2
Elements:  NumOfSells=3000
----------------------
Selling:
Attributes: SBookID=5, SBookstoreID=3
Elements:  NumOfSells=2500
----------------------
```

```
-------BOOKSTORE-------

Bookstore:
Attributes: BookstoreID=1
Elements: BookstoreName=Kinokuniya, StoreAddress: SCity=TokyoSStreet=3-chōme, ShinjukuSHouseNumber=7-17
----------------------
Bookstore:
Attributes: BookstoreID=2
Elements: BookstoreName=Daikanyama T-Site, StoreAddress: SCity=TokyoSStreet=Sarugaku-cho, ShibuyaSHouseNumber=17-5
----------------------
Bookstore:
Attributes: BookstoreID=3
Elements: BookstoreName=Book First Umeda, StoreAddress: SCity=OsakaSStreet=Kita-ku, ShibataSHouseNumber=1-1-3
----------------------
Disconnected from the target VM, address: '127.0.0.1:62980', transport: 'socket'

Process finished with exit code 0
```

2b) adatmódosítás

Adatmódosításnál miután létrehoztuk a kódban a dokumentumot és beolvastuk az adatokat, elkezdjük a belső módosításokat külön erre a célra megírt függvényekkel. Ezeknek a függvényeknek meg kell adni hogy melyik főbb elem hányadik példányának melyik gyerekelemét kívánjuk módosítani és hogy mire. Miután megtörtént a dokumentumon belül a módosítás egy Transformer és a FileOutput segítségével frissítjük a korábban beolvasott XML-t segítségével (ebben az esetben új fájlt hoztam létre, hogy látszódjon a különbség, de rá is lehetne menteni).

(DomModifyA7TIJX.java) ---

```java
package hu.domparse.a7tijx;

import hu.domparse.a7tijx.elements.*;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;

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 java.io.FileOutputStream;
import java.util.List;

import static hu.domparse.a7tijx.elements.Author.makeAuthorList;
import static hu.domparse.a7tijx.elements.Book.makeBookList;
import static hu.domparse.a7tijx.elements.Bookstore.makeBookstoreList;
import static hu.domparse.a7tijx.elements.Magazine.makeMagazineList;
import static hu.domparse.a7tijx.elements.President.makePresidentList;
import static hu.domparse.a7tijx.elements.Publisher.makePublisherList;
import static hu.domparse.a7tijx.elements.Selling.makeSellingList;
import static hu.domparse.a7tijx.elements.Series.makeSeriesList;

public class DomModifyA7TIJX {
    public static void main(String[] args) {
        try{
            //XML dokumentum read
            DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = builderFactory.newDocumentBuilder();
            Document document = builder.parse("XMLA7TIJX.xml");
            document.getDocumentElement().normalize();

            //XML dokumentum modify
            modifyElementXML(document, "Series", 0, "SeriesName", "Bomby Girl");
            modifyElementXML(document, "Magazine", 2, "PublishingFrequency", "Monthly");
            modifyElementXML(document, "Publisher", 1, "PhoneNumber", "123-4567-8910");
            modifyElementXML(document, "President", 0, "PresidentGender", "Female");
            modifyAttributeXML(document, "Selling", 4, "SBookID", "55");
            modifyAttributeXML(document, "Book", 3, "BSeries", "4");
```

```java
            try {
                //XML dokumentum write
                TransformerFactory transformerFactory = TransformerFactory.newInstance();
                Transformer transformer = transformerFactory.newTransformer();
                DOMSource domSource = new DOMSource(document);
                FileOutputStream output = new FileOutputStream("XMLA7TIJXmodify.xml");
                StreamResult streamResult = new StreamResult(output);
                transformer.transform(domSource, streamResult);

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

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

    }

    //Ha elemet akarunk módosítani
    public static void modifyElementXML(Document document, String mainTag, int mainTagItem, String targetTag, String modifiedString){
            for(int i = 0; i < document.getElementsByTagName(mainTag).item(mainTagItem).getChildNodes().getLength(); i++){
                if(document.getElementsByTagName(mainTag).item(mainTagItem).getChildNodes().item(i).getNodeName() == targetTag){
                    System.out.println("Before: " + document.getElementsByTagName(mainTag).item(mainTagItem).getChildNodes().item(i).getTextContent());
                    document.getElementsByTagName(mainTag).item(mainTagItem).getChildNodes().item(i).setTextContent(modifiedString);
                    System.out.println("After: " + document.getElementsByTagName(mainTag).item(mainTagItem).getChildNodes().item(i).getTextContent());
                }
            }
            System.out.println("----------------------");

    }

    //Ha attribútumot akarunk módasítani
    public static void modifyAttributeXML(Document document, String mainTag, int mainTagItem, String targetAttribute, String modifiedString){
            for(int i = 0; i < document.getElementsByTagName(mainTag).item(mainTagItem).getAttributes().getLength(); i++){
                if(document.getElementsByTagName(mainTag).item(mainTagItem).getAttributes().item(i).getNodeName() == targetAttribute){
                    System.out.println("Before: " + document.getElementsByTagName(mainTag).item(mainTagItem).getAttributes().item(i).getTextContent());
                    document.getElementsByTagName(mainTag).item(mainTagItem).getAttributes().item(i).setTextContent(modifiedString);
                    System.out.println("After: " + document.getElementsByTagName(mainTag).item(mainTagItem).getAttributes().item(i).getTextContent());
                }
            }
            System.out.println("----------------------");
    }

}
```

Ezek a módosított elemek:

```
<Series SAuthor="2" SMagazine="3" SeriesID="1">
    <SeriesName>Bomby Girl</SeriesName>
    <Ranking>3#</Ranking>
    <Genre>
        <Subgenre>Action</Subgenre>
        <Subgenre>Supernatural</Subgenre>
        <Subgenre>Gore</Subgenre>
    </Genre>
    <NumOfReaders>750000</NumOfReaders>
</Series>
```

```
    <Magazine MPublisher="1" MagazineID="3">
        <MagazineName>Weekly Shounen Jump</MagazineName>
        <PublishingFrequency>Monthly</PublishingFrequency>
        <Demographic>Shounen</Demographic>
        <MagazinePrice>¥250</MagazinePrice>
    </Magazine>
```

```
    <Publisher PPresident="3" PublisherID="2">
        <PublisherName>Shogakukan</PublisherName>
        <PhoneNumber>123-4567-8910</PhoneNumber>
        <EmailAddress>doe@shogakukan.co.jp</EmailAddress>
        <HQAddress>
            <PCity>Tokyo</PCity>
            <PStreet>Hitotsubashi Chiyoda-ku</PStreet>
            <PHouseNumber>2-3-1</PHouseNumber>
        </HQAddress>
    </Publisher>
```

```
    <President PresidentID="1">
        <PresidentName>Noma Yoshinobu</PresidentName>
        <PresidentGender>Female</PresidentGender>
        <PresidentAge>54</PresidentAge>
    </President>
```

```
<Selling SBookID="55" SBookstoreID="2">
    <NumOfSells>2500</NumOfSells>
</Selling>
```

```
<Book BSeries="4" BookID="4">
    <BookName>20th Century Boys Vol.2</BookName>
    <BookPrice>$30</BookPrice>
    <NumOfPages>340</NumOfPages>
    <NumOfChapters>11</NumOfChapters>
    <VolumeNumber>2</VolumeNumber>
</Book>
```

2c) adatlekérdezés

Az adatok lekérdezése az osztályokból származó előnyök miatt igen egyszerűen megvalósítható. Sima ciklusokkal át lehet futni és meg lehet keresni a kívánt elemeket amit további keresésekhez használhatunk, így összetett lekérdezéseket is megvalósíthatunk.

(DomQueryA7TIJX.java) ---

```java
package hu.domparse.a7tijx;

import hu.domparse.a7tijx.elements.*;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.util.ArrayList;
import java.util.List;

import static hu.domparse.a7tijx.elements.Author.makeAuthorList;
import static hu.domparse.a7tijx.elements.Book.makeBookList;
import static hu.domparse.a7tijx.elements.Bookstore.makeBookstoreList;
import static hu.domparse.a7tijx.elements.Magazine.makeMagazineList;
import static hu.domparse.a7tijx.elements.President.makePresidentList;
import static hu.domparse.a7tijx.elements.Publisher.makePublisherList;
import static hu.domparse.a7tijx.elements.Selling.makeSellingList;
import static hu.domparse.a7tijx.elements.Series.makeSeriesList;

public class DomQueryA7TIJX {
    public static void main(String[] args) {
        try{
            //XML dokumentum read
            DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = builderFactory.newDocumentBuilder();
            Document document = builder.parse("XMLA7TIJX.xml");
            document.getDocumentElement().normalize();

            //Node listák kiszedése a dokumentumból
            NodeList kkiado_a7tijx = document.getElementsByTagName("Kepregeny_kiado_A7TIJX");

            NodeList SeriesNodeList = document.getElementsByTagName("Series");
            NodeList AuthorNodeList = document.getElementsByTagName("Author");
            NodeList MagazineNodeList = document.getElementsByTagName("Magazine");
            NodeList PublisherNodeList = document.getElementsByTagName("Publisher");
            NodeList PresidentNodeList = document.getElementsByTagName("President");
            NodeList BookNodeList = document.getElementsByTagName("Book");
            NodeList SellingNodeList = document.getElementsByTagName("Selling");
            NodeList BookstoreNodeList = document.getElementsByTagName("Bookstore");

            //Node listákból elem listákat csinálunk
            List<Series> seriesList = makeSeriesList(SeriesNodeList);
            List<Author> authorList = makeAuthorList(AuthorNodeList);
            List<Magazine> magazineList = makeMagazineList(MagazineNodeList);
            List<Publisher> publisherList = makePublisherList(PublisherNodeList);
            List<President> presidentList = makePresidentList(PresidentNodeList);
            List<Book> bookList = makeBookList(BookNodeList);
            List<Selling> sellingList = makeSellingList(SellingNodeList);
            List<Bookstore> bookstoreList = makeBookstoreList(BookstoreNodeList);

```

```java
        //Lekérdezések

        //Attack on Titan képregény rankja
        System.out.println("Attack on Titan képregény rankja: ");
        for (Series s : seriesList) {
            if(s.seriesName.equals("Attack on Titan")){
                System.out.println(s.ranking);
            }
        }
        System.out.println("-------------");

        //A Sci-Fi műfajban jártas írók neve
        System.out.println("A Sci-Fi műfajban jártas írók neve: ");
        List<Integer> iID = new ArrayList<Integer>();
        for (Series s : seriesList) {
            if(s.subGenre.contains("Sci-fi")) {
                iID.add(s.sAuthor);
            }
        }
        for (Author a : authorList) {
            for (int i : iID) {
                if(a.authorID == i){
                    System.out.println(a.authorName);
                }
            }
        }
        System.out.println("-------------");

        //Book First Umeda könyvesboltban árult könyvek
        System.out.println("Book First Umeda könyvesboltban árult könyvek: ");
        int bsID = -1;
        List<Integer> bukID = new ArrayList<Integer>();
        for (Bookstore bs : bookstoreList) {
            if(bs.bookstoreName.equals("Book First Umeda")){
                bsID = bs.bookstoreID;
            }
        }
        for (Selling sell : sellinglist) {
            if(sell.sBookstoreID == bsID){
                bukID.add(sell.sBookID);
            }
        }
        for (Book b : bookList) {
            for (int i : bukID) {
                if(b.bookID == i){
                    System.out.println(b.bookName);
                }
            }
        }
        System.out.println("-------------");
```

```java
        //Urasawa Naoki kiadójának a teljes címe
        System.out.println("Urasawa Naoki kiadójának a teljes címe: ");
        int uID = -1, mID = -1;
        for (Author auth : authorList) {
            if(auth.authorName.equals("Urasawa Naoki")){
                uID = auth.authorID;
            }
        }
        for (Series si : seriesList) {
            if(si.sAuthor == uID){
                mID = si.sMagazine;
            }
        }
        for (Magazine m : magazineList) {
            if(m.magazineID == mID){
                for (Publisher p : publisherList){
                    if(m.mPublisher == p.publisherID){
                        System.out.println(p.publisherName);
                        System.out.println(p.pCity);
                        System.out.println(p.pStreet);
                        System.out.println(p.pHouseNumber);
                    }
                }
            }
        }
        System.out.println("-------------");

        //Azoknak a könyveknek a nevének a lekérdezése amelyeket Fujimoto Tatsuki írt
        int aID = -1;
        List<Integer> sID = new ArrayList<Integer>();
        for (Author auth : authorList) {
            if(auth.authorName.equals("Fujimoto Tatsuki")){
                aID = auth.authorID;
            }
        }
        for (Series seri : seriesList) {
            if(seri.sAuthor == aID){
                sID.add(seri.seriesID);
            }
        }
        System.out.println("Fujimoto Tatsuki által készített könyvek:");
        for(Book book : bookList) {
            for (int s : sID) {
                if(book.bSeries == s){
                    System.out.println(book.bookName);
                }
            }
        }
        System.out.println("-------------");
    }
    catch (Exception ee) {
        ee.printStackTrace();
    }
```

A lekérdezések eredménye:

```
Attack on Titan képregény rankja:
2#
-------------
A Sci-Fi műfajban jártas írók neve:
Urasawa Naoki
Fujimoto Tatsuki
-------------
Book First Umeda könyvesboltban árult könyvek:
Chainsaw Man Vol.1
Chainsaw Man Vol.2
Attack on Titan Vol.1
Fire Punch Vol.1
-------------
Urasawa Naoki kiadójának a teljes címe:
Shogakukan
Tokyo
Hitotsubashi Chiyoda-ku
2-3-1
-------------
Fujimoto Tatsuki által készített könyvek:
Chainsaw Man Vol.1
Chainsaw Man Vol.2
Fire Punch Vol.1
-------------
```

2d) adatírás

Az adatok kiírása szintén a Transformerrel történik egyszerű módon, akárcsak a modify esetében. Betápláltam a dokumentumot majd az felhasználva létrehoztam egy új fájlt.

(DomWriteA7TIJX.java) ---

```java
package hu.domparse.a7tijx;

import org.w3c.dom.Document;

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 java.io.FileOutputStream;

public class DOMWriteA7TIJX {
    public static void main(String[] args) {
        try{
            //XML dokumentum read
            DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = builderFactory.newDocumentBuilder();
            Document document = builder.parse("XMLA7TIJX.xml");
            document.getDocumentElement().normalize();

            try {
                //XML dokumentum write
                TransformerFactory transformerFactory = TransformerFactory.newInstance();
                Transformer transformer = transformerFactory.newTransformer();
                DOMSource domSource = new DOMSource(document);
                FileOutputStream output = new FileOutputStream("XMLA7TIJX1.xml");
                StreamResult streamResult = new StreamResult(output);
                transformer.transform(domSource, streamResult);
                transformer.transform(domSource, new StreamResult(System.out));

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

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

    }
}

```