

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 képregénypiac, 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,

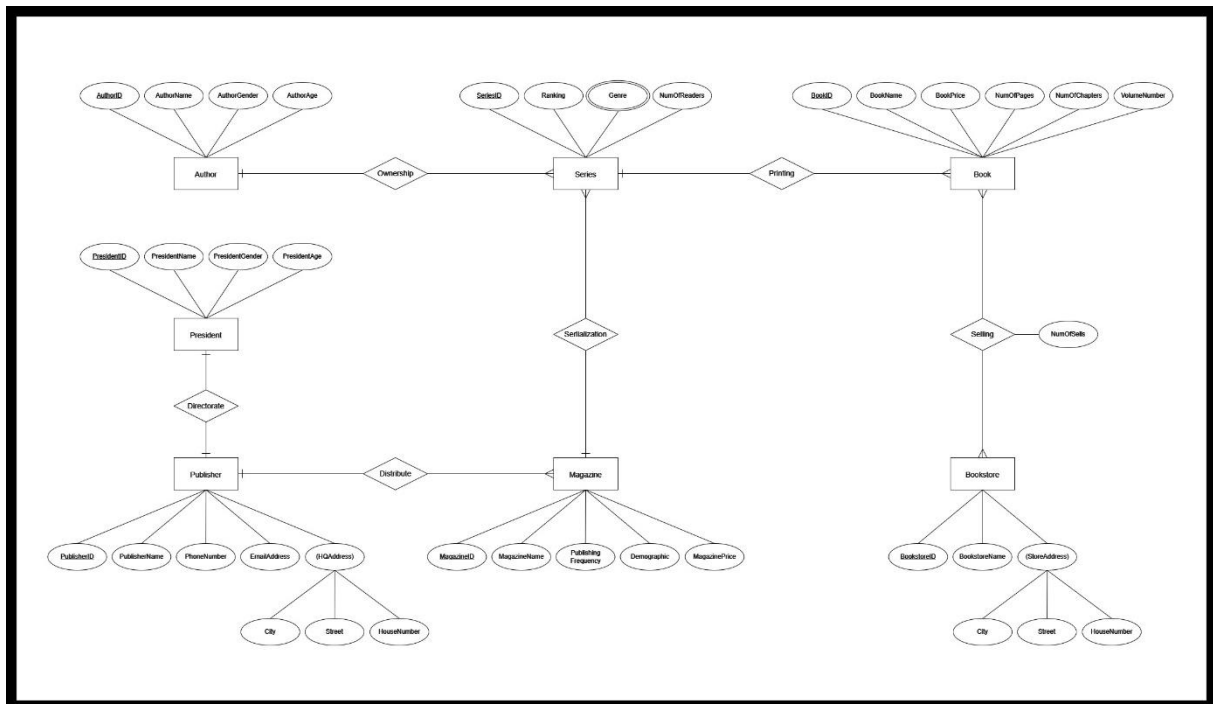
a seinen felnőtt férfiaknak, a shojo fiatal lányoknak és a josei felnőtt nőknek készül. Minden kiadónak van egy elnöke aki főként hozza a döntéseket a kiadóval kapcsolatban. Miután összegyűlt egy adag (általában úgy 200-250 oldalnyi) fejezet, ezeket könyvekbe gyűjtik és így kiadják. Ezeket tudják venni 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.

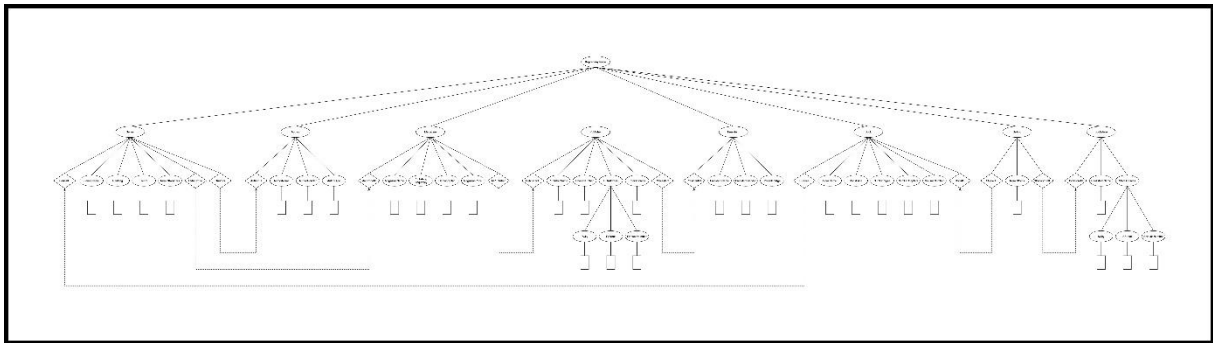
Az Author a képregényszerző, a Series a képregényszériák amiket az szerzők alkotnak és a magazinokban jelennek meg. A magazin (Magazine) a kiadó által vezetett nyomtatott képregények gyűjteménye ami adott időközönként jelenik meg. A kiadó (Publisher) amely üzemelteti a magazint, valamint ennek van egy 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 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 IDjét 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) ---

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?><Képregeny_kiado_A7TIJX>
2
3 <!--A jelenleg futó képregényszériák-->
4
5 <Series SAuthor="2" SMagazine="3" SeriesID="1">
6   <SeriesName>Chainsaw Man</SeriesName>
7   <Ranking>3#</Ranking>
8   <Genre>
9     <Subgenre>Action</Subgenre>
10    <Subgenre>Supernatural</Subgenre>
11    <Subgenre>Gore</Subgenre>
12  </Genre>
13  <NumOfReaders>750000</NumOfReaders>
14 </Series>
15
16 <Series SAuthor="1" SMagazine="2" SeriesID="2">
17   <SeriesName>20th Century Boys</SeriesName>
18   <Ranking>23#</Ranking>
19   <Genre>
20     <Subgenre>Drama</Subgenre>
21     <Subgenre>Mystery</Subgenre>
22     <Subgenre>Sci-fi</Subgenre>
23   </Genre>
24   <NumOfReaders>340000</NumOfReaders>
25 </Series>
26
27 <Series SAuthor="3" SMagazine="1" SeriesID="3">
28   <SeriesName>Attack on Titan</SeriesName>
29   <Ranking>2#</Ranking>
30   <Genre>
31     <Subgenre>Action</Subgenre>
32     <Subgenre>Drama</Subgenre>
33     <Subgenre>Suspense</Subgenre>
34   </Genre>
35   <NumOfReaders>2700000</NumOfReaders>
36 </Series>
37
38 <Series SAuthor="2" SMagazine="4" SeriesID="4">
39   <SeriesName>Fire Punch</SeriesName>
40   <Ranking>59#</Ranking>
41   <Genre>
42     <Subgenre>Action</Subgenre>
43     <Subgenre>Supernatural</Subgenre>
44     <Subgenre>Sci-fi</Subgenre>
45   </Genre>
46   <NumOfReaders>85000</NumOfReaders>
47 </Series>
48
```

```

49 <!--A képregények szerzői-->
50
51 <Author AuthorID="1">
52   <AuthorName>Urasawa Naoki</AuthorName>
53   <AuthorGender>Male</AuthorGender>
54   <AuthorAge>63</AuthorAge>
55 </Author>
56
57 <Author AuthorID="2">
58   <AuthorName>Fujimoto Tatsuki</AuthorName>
59   <AuthorGender>Male</AuthorGender>
60   <AuthorAge>31</AuthorAge>
61 </Author>
62
63 <Author AuthorID="3">
64   <AuthorName>Isayama Hajime</AuthorName>
65   <AuthorGender>Male</AuthorGender>
66   <AuthorAge>37</AuthorAge>
67 </Author>
68
69 <!--A magazinok amelyekben szereplenek a képregények-->
70
71 <Magazine MPublisher="3" MagazineID="1">
72   <MagazineName>Monthly Shounen Magazine</MagazineName>
73   <PublishingFrequency>Monthly</PublishingFrequency>
74   <Demographic>Shounen</Demographic>
75   <MagazinePrice>¥270</MagazinePrice>
76 </Magazine>
77
78 <Magazine MPublisher="2" MagazineID="2">
79   <MagazineName>Big Comic Spirits</MagazineName>
80   <PublishingFrequency>Weekly</PublishingFrequency>
81   <Demographic>Seinen</Demographic>
82   <MagazinePrice>¥280</MagazinePrice>
83 </Magazine>
84
85 <Magazine MPublisher="1" MagazineID="3">
86   <MagazineName>Weekly Shounen Jump</MagazineName>
87   <PublishingFrequency>Weekly</PublishingFrequency>
88   <Demographic>Shounen</Demographic>
89   <MagazinePrice>¥250</MagazinePrice>
90 </Magazine>
91
92 <Magazine MPublisher="1" MagazineID="4">
93   <MagazineName>Shounen Jump+</MagazineName>
94   <PublishingFrequency>Weekly</PublishingFrequency>
95   <Demographic>Shounen</Demographic>
96   <MagazinePrice>¥200</MagazinePrice>
97 </Magazine>

```



```

99      <!--A kiadók amelyek kiadják a magazinokat-->
100
101      <Publisher PPresident="2" PublisherID="1">
102          <PublisherName>Shueisha</PublisherName>
103          <PhoneNumber>090-1731-1447</PhoneNumber>
104          <EmailAddress>doe@shueisha.co.jp</EmailAddress>
105          <HQAddress>
106              <PCity>Tokyo</PCity>
107              <PStreet>HitotsubashivChiyoda-ku</PStreet>
108              <PHouseNumber>2-5-10</PHouseNumber>
109          </HQAddress>
110      </Publisher>
111
112      <Publisher PPresident="3" PublisherID="2">
113          <PublisherName>Shogakukan</PublisherName>
114          <PhoneNumber>070-6386-1759</PhoneNumber>
115          <EmailAddress>doe@shogakukan.co.jp</EmailAddress>
116          <HQAddress>
117              <PCity>Tokyo</PCity>
118              <PStreet>Hitotsubashi Chiyoda-ku</PStreet>
119              <PHouseNumber>2-3-1</PHouseNumber>
120          </HQAddress>
121      </Publisher>
122
123      <Publisher PPresident="1" PublisherID="3">
124          <PublisherName>Kodansha</PublisherName>
125          <PhoneNumber>080-9598-3865</PhoneNumber>
126          <EmailAddress>doe@kodansha.co.jp</EmailAddress>
127          <HQAddress>
128              <PCity>Tokyo</PCity>
129              <PStreet>Otowa Bunkyo-ku </PStreet>
130              <PHouseNumber>2-12-21</PHouseNumber>
131          </HQAddress>
132      </Publisher>
133
134      <!--A kiadók elnökei-->
135
136      <President PresidentID="1">
137          <PresidentName>Noma Yoshinobu</PresidentName>
138          <PresidentGender>Male</PresidentGender>
139          <PresidentAge>54</PresidentAge>
140      </President>
141
142      <President PresidentID="2">
143          <PresidentName>Horiuchi Marue</PresidentName>
144          <PresidentGender>Male</PresidentGender>
145          <PresidentAge>72</PresidentAge>
146      </President>
147
148      <President PresidentID="3">
149          <PresidentName>Oga Nobuhiro</PresidentName>
150          <PresidentGender>Male</PresidentGender>
151          <PresidentAge>75</PresidentAge>
152      </President>
153

```

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

```
155
```

```
156 <Book BSeries="1" BookID="1">
```

```
157   <BookName>Chainsaw Man Vol.1</BookName>
```

```
158   <BookPrice>$20</BookPrice>
```

```
159   <NumOfPages>200</NumOfPages>
```

```
160   <NumOfChapters>8</NumOfChapters>
```

```
161   <VolumeNumber>1</VolumeNumber>
```

```
162 </Book>
```

```
163
```

```
164 <Book BSeries="1" BookID="2">
```

```
165   <BookName>Chainsaw Man Vol.2</BookName>
```

```
166   <BookPrice>$20</BookPrice>
```

```
167   <NumOfPages>200</NumOfPages>
```

```
168   <NumOfChapters>7</NumOfChapters>
```

```
169   <VolumeNumber>2</VolumeNumber>
```

```
170 </Book>
```

```
171
```

```
172 <Book BSeries="3" BookID="3">
```

```
173   <BookName>Attack on Titan Vol.1</BookName>
```

```
174   <BookPrice>$25</BookPrice>
```

```
175   <NumOfPages>230</NumOfPages>
```

```
176   <NumOfChapters>7</NumOfChapters>
```

```
177   <VolumeNumber>1</VolumeNumber>
```

```
178 </Book>
```

```
179
```

```
180 <Book BSeries="2" BookID="4">
```

```
181   <BookName>20th Century Boys Vol.2</BookName>
```

```
182   <BookPrice>$30</BookPrice>
```

```
183   <NumOfPages>340</NumOfPages>
```

```
184   <NumOfChapters>11</NumOfChapters>
```

```
185   <VolumeNumber>2</VolumeNumber>
```

```
186 </Book>
```

```
187
```

```
188 <Book BSeries="4" BookID="5">
```

```
189   <BookName>Fire Punch Vol.1</BookName>
```

```
190   <BookPrice>$25</BookPrice>
```

```
191   <NumOfPages>260</NumOfPages>
```

```
192   <NumOfChapters>9</NumOfChapters>
```

```
193   <VolumeNumber>1</VolumeNumber>
```

```
194 </Book>
```

```
195
```

```

196 <!--Eladások a könyvesboltokban-->
197
198 <Selling SBookID="1" SBookstoreID="1">
199   <NumOfSells>4000</NumOfSells>
200 </Selling>
201
202 <Selling SBookID="1" SBookstoreID="2">
203   <NumOfSells>3000</NumOfSells>
204 </Selling>
205
206 <Selling SBookID="1" SBookstoreID="3">
207   <NumOfSells>7000</NumOfSells>
208 </Selling>
209
210 <Selling SBookID="2" SBookstoreID="1">
211   <NumOfSells>3000</NumOfSells>
212 </Selling>
213
214 <Selling SBookID="2" SBookstoreID="2">
215   <NumOfSells>2500</NumOfSells>
216 </Selling>
217
218 <Selling SBookID="2" SBookstoreID="3">
219   <NumOfSells>4000</NumOfSells>
220 </Selling>
221
222 <Selling SBookID="3" SBookstoreID="1">
223   <NumOfSells>8000</NumOfSells>
224 </Selling>
225
226 <Selling SBookID="3" SBookstoreID="3">
227   <NumOfSells>10000</NumOfSells>
228 </Selling>
229
230 <Selling SBookID="4" SBookstoreID="2">
231   <NumOfSells>3000</NumOfSells>
232 </Selling>
233
234 <Selling SBookID="5" SBookstoreID="3">
235   <NumOfSells>2500</NumOfSells>
236 </Selling>
237

```

```

238      <!--A könyvesboltok ahol eladják könyveket-->
239
240      <Bookstore BookstoreID="1">
241        <BookstoreName>Kinokuniya</BookstoreName>
242        <StoreAddress>
243          <SCity>Tokyo</SCity>
244          <SStreet>3-chōme, Shinjuku</SStreet>
245          <SHouseNumber>7-17</SHouseNumber>
246        </StoreAddress>
247      </Bookstore>
248
249      <Bookstore BookstoreID="2">
250        <BookstoreName>Daikanyama T-Site</BookstoreName>
251        <StoreAddress>
252          <SCity>Tokyo</SCity>
253          <SStreet>Sarugaku-cho, Shibuya</SStreet>
254          <SHouseNumber>17-5</SHouseNumber>
255        </StoreAddress>
256      </Bookstore>
257
258      <Bookstore BookstoreID="3">
259        <BookstoreName>Book First Umeda</BookstoreName>
260        <StoreAddress>
261          <SCity>Osaka</SCity>
262          <SStreet>Kita-ku, Shibata</SStreet>
263          <SHouseNumber>1-1-3</SHouseNumber>
264        </StoreAddress>
265      </Bookstore>
266
267    </Kepregeny_kiado_A7TIJX>

```

1d) Az XML dokumentum alapján XMLSchema készítése

Az XSD fájl az XML fájl létrehozásának segítségével 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) ---

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3
4      <!--Képregény kiadó-->
5
6      <xs:element name="Kepregeny_kiado_A7TIJX">
7          <xs:complexType>
8              <xs:sequence>
9                  <xs:element name="Series" type="SeriesType" minOccurs="1" maxOccurs="unbounded"/>
10                 <xs:element name="Author" type="AuthorType" minOccurs="1" maxOccurs="unbounded"/>
11                 <xs:element name="Magazine" type="MagazineTypes" minOccurs="1" maxOccurs="unbounded"/>
12                 <xs:element name="Publisher" type="PublisherType" minOccurs="1" maxOccurs="unbounded"/>
13                 <xs:element name="President" type="PresidentType" minOccurs="1" maxOccurs="unbounded"/>
14                 <xs:element name="Book" type="BookType" minOccurs="1" maxOccurs="unbounded"/>
15                 <xs:element name="Selling" type="SellingType" minOccurs="1" maxOccurs="unbounded"/>
16                 <xs:element name="Bookstore" type="BookstoreType" minOccurs="1" maxOccurs="unbounded"/>
17             </xs:sequence>
18         </xs:complexType>
19     </xs:element>
20
21     <!--Főbb komplex típusok-->
22
23     <xs:complexType name="SeriesType">
24         <xs:sequence>
25             <xs:element ref="SeriesName" minOccurs="1" maxOccurs="1"/>
26             <xs:element ref="Ranking" minOccurs="1" maxOccurs="1"/>
27             <xs:element ref="Genre" minOccurs="1" maxOccurs="1"/>
28             <xs:element ref="NumOfReaders" minOccurs="1" maxOccurs="1"/>
29         </xs:sequence>
30         <xs:attribute name="SeriesID" type="xs:integer" use="required"/>
31         <xs:attribute name="SMagazine" type="xs:integer" use="required"/>
32         <xs:attribute name="SAuthor" type="xs:integer" use="required"/>
33     </xs:complexType>
34
35     <xs:complexType name="AuthorType">
36         <xs:sequence>
37             <xs:element ref="AuthorName" minOccurs="1" maxOccurs="1"/>
38             <xs:element ref="AuthorGender" minOccurs="1" maxOccurs="1"/>
39             <xs:element ref="AuthorAge" minOccurs="1" maxOccurs="1"/>
40         </xs:sequence>
41         <xs:attribute name="AuthorID" type="xs:integer" use="required"/>
42     </xs:complexType>
43
44     <xs:complexType name="MagazineTypes">
45         <xs:sequence>
46             <xs:element ref="MagazineName" minOccurs="1" maxOccurs="1"/>
47             <xs:element ref="PublishingFrequency" minOccurs="1" maxOccurs="1"/>
48             <xs:element ref="Demographic" minOccurs="1" maxOccurs="1"/>
49             <xs:element ref="MagazinePrice" minOccurs="1" maxOccurs="1"/>
50         </xs:sequence>
51         <xs:attribute name="MagazineID" type="xs:integer" use="required"/>
52         <xs:attribute name="MPublisher" type="xs:integer" use="required"/>
53     </xs:complexType>
```

```

55 <xs:complexType name="PublisherType">
56   <xs:sequence>
57     <xs:element ref="PublisherName" minOccurs="1" maxOccurs="1"/>
58     <xs:element ref="PhoneNumber" minOccurs="1" maxOccurs="1"/>
59     <xs:element ref="EmailAddress" minOccurs="1" maxOccurs="1"/>
60     <xs:element ref="HQAddress" minOccurs="1" maxOccurs="1"/>
61   </xs:sequence>
62   <xs:attribute name="PublisherID" type="xs:integer" use="required"/>
63   <xs:attribute name="PPresident" type="xs:integer" use="required"/>
64 </xs:complexType>
65
66 <xs:complexType name="PresidentType">
67   <xs:sequence>
68     <xs:element ref="PresidentName" minOccurs="1" maxOccurs="1"/>
69     <xs:element ref="PresidentGender" minOccurs="1" maxOccurs="1"/>
70     <xs:element ref="PresidentAge" minOccurs="1" maxOccurs="1"/>
71   </xs:sequence>
72   <xs:attribute name="PresidentID" type="xs:integer" use="required"/>
73 </xs:complexType>
74
75 <xs:complexType name="BookType">
76   <xs:sequence>
77     <xs:element ref="BookName" minOccurs="1" maxOccurs="1"/>
78     <xs:element ref="BookPrice" minOccurs="1" maxOccurs="1"/>
79     <xs:element ref="NumOfPages" minOccurs="1" maxOccurs="1"/>
80     <xs:element ref="NumOfChapters" minOccurs="1" maxOccurs="1"/>
81     <xs:element ref="VolumeNumber" minOccurs="1" maxOccurs="1"/>
82   </xs:sequence>
83   <xs:attribute name="BookID" type="xs:integer" use="required"/>
84   <xs:attribute name="BSeries" type="xs:integer" use="required"/>
85 </xs:complexType>
86
87 <xs:complexType name="SellingType">
88   <xs:sequence>
89     <xs:element name="NumOfSells" minOccurs="1" maxOccurs="1"/>
90   </xs:sequence>
91   <xs:attribute name="SBookID" type="xs:integer" use="required"/>
92   <xs:attribute name="SBookstoreID" type="xs:integer" use="required"/>
93 </xs:complexType>
94
95 <xs:complexType name="BookstoreType">
96   <xs:sequence>
97     <xs:element ref="BookstoreName" minOccurs="1" maxOccurs="1"/>
98     <xs:element ref="StoreAddress" minOccurs="1" maxOccurs="1"/>
99   </xs:sequence>
100   <xs:attribute name="BookstoreID" type="xs:integer" use="required"/>
101 </xs:complexType>
102

```

```

103 <!--Egyszerű típusok-->
104
105 <!--Series-->
106 <xs:element name="SeriesName" type="xs:string"/>
107 <xs:element name="Ranking" type="RankingType"/>
108 <xs:element name="NumOfReaders" type="xs:int"/>
109 <xs:element name="Genre" type="GenreType"/>
110
111 <xs:element name="Subgenre" type="xs:string"/>
112
113 <!--Author-->
114 <xs:element name="AuthorName" type="xs:string"/>
115 <xs:element name="AuthorAge" type="xs:int"/>
116 <xs:element name="AuthorGender" type="AuthorGenderType"/>
117
118 <!--Magazine-->
119 <xs:element name="MagazineName" type="xs:string"/>
120 <xs:element name="PublishingFrequency" type="PublishingFrequencyType"/>
121 <xs:element name="Demographic" type="DemographicType"/>
122 <xs:element name="MagazinePrice" type="MagazinePriceType"/>
123
124 <!--Publisher-->
125 <xs:element name="PublisherName" type="xs:string"/>
126 <xs:element name="PhoneNumber" type="PhoneNumberType"/>
127 <xs:element name="EmailAddress" type="EmailAddressType"/>
128 <xs:element name="HQAddress" type="HQAddressType"/>
129
130 <xs:element name="PCity" type="xs:string"/>
131 <xs:element name="PStreet" type="xs:string"/>
132 <xs:element name="PHouseNumber" type="xs:string"/>
133
134 <!--President-->
135 <xs:element name="PresidentName" type="xs:string"/>
136 <xs:element name="PresidentAge" type="xs:int"/>
137 <xs:element name="PresidentGender" type="PresidentGenderType"/>
138
139 <!--Book-->
140 <xs:element name="BookName" type="xs:string"/>
141 <xs:element name="BookPrice" type="BookPriceType"/>
142 <xs:element name="NumOfPages" type="xs:int"/>
143 <xs:element name="NumOfChapters" type="xs:int"/>
144 <xs:element name="VolumeNumber" type="xs:int"/>
145
146 <!--Selling-->
147 <xs:element name="NumOfSells" type="xs:int"/>
148
149 <!--Bookstore-->
150 <xs:element name="BookstoreName" type="xs:string"/>
151 <xs:element name="StoreAddress" type="StoreAddressType"/>
152
153 <xs:element name="SCity" type="xs:string"/>
154 <xs:element name="SStreet" type="xs:string"/>
155 <xs:element name="SHouseNumber" type="xs:string"/>
156

```

```

157 <!--Saját típusok-->
158
159 <xs:simpleType name="RankingType">
160   <xs:restriction base="xs:string">
161     <xs:pattern value="([0-9])+#" />
162   </xs:restriction>
163 </xs:simpleType>
164
165 <xs:simpleType name="AuthorGenderType">
166   <xs:restriction base="xs:string">
167     <xs:enumeration value="Male" />
168     <xs:enumeration value="Female" />
169   </xs:restriction>
170 </xs:simpleType>
171
172 <xs:simpleType name="PublishingFrequencyType">
173   <xs:restriction base="xs:string">
174     <xs:enumeration value="Weekly" />
175     <xs:enumeration value="Bi-Weekly" />
176     <xs:enumeration value="Monthly" />
177     <xs:enumeration value="Quarterly" />
178   </xs:restriction>
179 </xs:simpleType>
180
181 <xs:simpleType name="DemographicType">
182   <xs:restriction base="xs:string">
183     <xs:enumeration value="Shounen" />
184     <xs:enumeration value="Seinen" />
185     <xs:enumeration value="Shojo" />
186     <xs:enumeration value="Josei" />
187   </xs:restriction>
188 </xs:simpleType>
189
190 <xs:simpleType name="MagazinePriceType">
191   <xs:restriction base="xs:string">
192     <xs:pattern value="¥([0-9])+"/>
193   </xs:restriction>
194 </xs:simpleType>
195
196 <xs:simpleType name="PhoneNumberType">
197   <xs:restriction base="xs:string">
198     <xs:pattern value="[0-9]{3}-[0-9]{4}-[0-9]{4}" />
199   </xs:restriction>
200 </xs:simpleType>
201
202 <xs:simpleType name="EmailAddressType">
203   <xs:restriction base="xs:string">
204     <xs:pattern value="^[^@]+@[^\.]+\.\.+" />
205   </xs:restriction>
206 </xs:simpleType>
207
208 <xs:simpleType name="PresidentGenderType">
209   <xs:restriction base="xs:string">
210     <xs:pattern value="Male|Female" />
211   </xs:restriction>
212 </xs:simpleType>
213
214 <xs:simpleType name="BookPriceType">
215   <xs:restriction base="xs:string">
216     <xs:pattern value="$([0-9])+"/>
217   </xs:restriction>
218 </xs:simpleType>

```



```

220 <!--Komplex típusok-->
221
222 <xs:complexType name="GenreType">
223   <xs:sequence>
224     <xs:element ref="Subgenre" minOccurs="1" maxOccurs="unbounded"/>
225   </xs:sequence>
226 </xs:complexType>
227
228 <xs:complexType name="HQAddressType">
229   <xs:sequence>
230     <xs:element ref="PCity" minOccurs="1" maxOccurs="1"/>
231     <xs:element ref="PStreet" minOccurs="1" maxOccurs="1"/>
232     <xs:element ref="PHouseNumber" minOccurs="1" maxOccurs="1"/>
233   </xs:sequence>
234 </xs:complexType>
235
236 <xs:complexType name="StoreAddressType">
237   <xs:sequence>
238     <xs:element ref="SCity" minOccurs="1" maxOccurs="1"/>
239     <xs:element ref="SStreet" minOccurs="1" maxOccurs="1"/>
240     <xs:element ref="SHouseNumber" minOccurs="1" maxOccurs="1"/>
241   </xs:sequence>
242 </xs:complexType>
243
244 </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 ---

```

1 package hu.domparse.a7tjix;
2 import java.util.List;
3 import javax.xml.parsers.DocumentBuilder;
4 import javax.xml.parsers.DocumentBuilderFactory;
5
6 import org.w3c.dom.*;
7
8 import hu.domparse.a7tjix.elements.*;
9
10 import static hu.domparse.a7tjix.elements.Author.makeAuthorList;
11 import static hu.domparse.a7tjix.elements.Book.makeBookList;
12 import static hu.domparse.a7tjix.elements.Bookstore.makeBookstoreList;
13 import static hu.domparse.a7tjix.elements.Magazine.makeMagazineList;
14 import static hu.domparse.a7tjix.elements.President.makePresidentList;
15 import static hu.domparse.a7tjix.elements.Publisher.makePublisherList;
16 import static hu.domparse.a7tjix.elements.Selling.makeSellingList;
17 import static hu.domparse.a7tjix.elements.Series.makeSeriesList;
18
19 public class DomReadA7TJIX {
20     public static void main(String[] args) {
21         try {
22             //XML dokumentum read
23             DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
24             DocumentBuilder builder = builderFactory.newDocumentBuilder();
25             Document document = builder.parse("70MLA7TJIX.xml");
26             document.getDocumentElement().normalize();
27
28             //Node listák kizsédése a dokumentumból
29             NodeList kklado_a7tjix = document.getElementsByTagName("Kepregeny_kiado_A7TJIX");
30
31             NodeList SeriesNodeList = document.getElementsByTagName("Series");
32             NodeList AuthorNodeList = document.getElementsByTagName("Author");
33             NodeList MagazineNodeList = document.getElementsByTagName("Magazine");
34             NodeList PublisherNodeList = document.getElementsByTagName("Publisher");
35             NodeList PresidentNodeList = document.getElementsByTagName("President");
36             NodeList BookNodeList = document.getElementsByTagName("Book");
37             NodeList SellingNodeList = document.getElementsByTagName("Selling");
38             NodeList BookstoreNodeList = document.getElementsByTagName("Bookstore");
39
40             //Node listákból elem listákat csinálunk
41             List<Series> seriesList = makeSeriesList(SeriesNodeList);
42             List<Author> authorList = makeAuthorList(AuthorNodeList);
43             List<Magazine> magazineList = makeMagazineList(MagazineNodeList);
44             List<Publisher> publisherList = makePublisherList(PublisherNodeList);
45             List<President> presidentList = makePresidentList(PresidentNodeList);
46             List<Book> bookList = makeBookList(BookNodeList);
47             List<Selling> sellingList = makeSellingList(SellingNodeList);
48             List<Bookstore> bookstoreList = makeBookstoreList(BookstoreNodeList);
49
50             //Teljes dokumentum kiírása egyedi strukturált formában listáként
51
52             System.out.println("\n-----SERIES-----\n");
53
54             for (Series s : seriesList) {
55                 System.out.println("Series:");
56                 System.out.println("Attributes: SeriesID=" + s.seriesID + ", SMagazine=" + s.sMagazine + ", SAuthor=" + s.sAuthor);
57                 System.out.println("Elements: SeriesName=" + s.seriesName + ", Ranking=" + s.ranking + ", NumOfReaders=" + s.numOfReaders + ", Genres: ");
58                 for (String subgenre : s.subGenre) {
59                     System.out.print("Subgenre=" + subgenre + " ");
60                 }
61                 System.out.println("\n");
62                 System.out.println("-----");
63             }
64
65             System.out.println("\n-----AUTHOR-----\n");
66
67             for (Author a : authorList) {
68                 System.out.println("Author:");
69                 System.out.println("Attributes: AuthorID=" + a.authorID);
70                 System.out.println("Elements: AuthorName=" + a.authorName + ", AuthorGender=" + a.authorGender + ", AuthorAge=" + a.authorAge);
71                 System.out.println("-----");
72             }
73
74             System.out.println("\n-----MAGAZINE-----\n");
75
76             for (Magazine m : magazineList) {
77                 System.out.println("Magazine:");
78                 System.out.println("Attributes: MagazineID=" + m.magazineID + ", MPublisher=" + m.mPublisher);
79                 System.out.println("Elements: MagazineName=" + m.magazineName + ", PublishingFrequency=" + m.publishingFrequency + ", Demographic=" + m.demographic + ", MagazinePrice=" + m.magazinePrice);
80                 System.out.println("-----");
81             }
82
83             System.out.println("\n-----PUBLISHER-----\n");
84
85             for (Publisher pu : publisherList) {
86                 System.out.println("Publisher:");
87                 System.out.println("Attributes: PublisherID=" + pu.publisherID + ", PPresident=" + pu.pPresident);
88                 System.out.println("Elements: PublisherName=" + pu.publisherName + ", PhoneNumber=" + pu.phoneNumber + ", EmailAddress=" + pu.emailAddress + ", HQAddress: PCity=" + pu.pCity + ", PStreet=" + pu.pStreet + "
89                 System.out.println("-----");
90             }
91
92             System.out.println("\n-----PRESIDENT-----\n");
93
94             for (President pr : presidentList) {
95                 System.out.println("President:");
96                 System.out.println("Attributes: PresidentID=" + pr.presidentID);
97                 System.out.println("Elements: PresidentName=" + pr.presidentName + ", PresidentGender=" + pr.presidentGender + ", PresidentAge=" + pr.presidentAge);
98                 System.out.println("-----");
99             }
100
101             System.out.println("\n-----PRESIDENT-----\n");
102
103             for (President pr : presidentList) {
104                 System.out.println("President:");
105                 System.out.println("Attributes: PresidentID=" + pr.presidentID);
106                 System.out.println("Elements: PresidentName=" + pr.presidentName + ", PresidentGender=" + pr.presidentGender + ", PresidentAge=" + pr.presidentAge);
107                 System.out.println("-----");
108             }
109
110             System.out.println("\n-----BOOK-----\n");
111
112             for (Book b : bookList) {
113                 System.out.println("Book:");
114                 System.out.println("Attributes: BookID=" + b.bookID + ", BSeries=" + b.bSeries);
115                 System.out.println("Elements: BookName=" + b.bookName + ", BookPrice=" + b.bookPrice + ", NumOfPages=" + b.numOfPages + ", NumOfChapters=" + b.numOfChapters + ", VolumeNumber=" + b.volumeNumber);
116                 System.out.println("-----");
117             }
118
119             System.out.println("\n-----SELLING-----\n");
120
121             for (Selling sl : sellingList) {
122                 System.out.println("Selling:");
123                 System.out.println("Attributes: SBookID=" + sl.sBookID + ", SBookstoreID=" + sl.sBookstoreID);
124                 System.out.println("Elements: NumOfSells=" + sl.numOfSells);
125                 System.out.println("-----");
126             }
127
128             System.out.println("\n-----BOOKSTORE-----\n");
129
130             for (Bookstore bs : bookstoreList) {
131                 System.out.println("Bookstore:");
132                 System.out.println("Attributes: BookstoreID=" + bs.bookstoreID);
133                 System.out.println("Elements: BookstoreName=" + bs.bookstoreName + ", StoreAddress: SCity=" + bs.sCity + "SStreet=" + bs.sStreet + "HouseNumber=" + bs.houseNumber);
134                 System.out.println("-----");
135             }
136
137             catch (Exception ee) {
138                 ee.printStackTrace();
139             }
140         }
141     }
142 }

```

Aktiválja a Windowst
Aktiválja a Windows rendszert a Gépén

Az itt listákhoz felhasznált osztályok a következőket tartalmazzák:

```
1 package hu.domparsa.a7tjix.elements;
2
3 import org.w3c.dom.NodeList;
4
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class Author {
9     //Attribútumok
10    public int authorID;
11
12    //Elemek
13    public String authorName;
14    public String authorGender;
15    public int authorAge;
16
17    //Author példány feltöltésére szolgál (segédfüggvény)
18    public static Author authorLoad(NodeList authorNodes, int item){
19        Author author = new Author();
20
21        //Feltölti az elemeket
22        for(int i = 0; i < authorNodes.item(item).getChildNodes().getLength(); i++){
23            String nodeName = authorNodes.item(item).getChildNodes().item(i).getNodeName();
24            String nodeContent = authorNodes.item(item).getChildNodes().item(i).getTextContent();
25            if(nodeName.equals("AuthorName")){author.authorName = nodeContent;}
26            else if(nodeName.equals("AuthorGender")){author.authorGender = nodeContent;}
27            else if(nodeName.equals("AuthorAge")){author.authorAge = Integer.parseInt(nodeContent);}
28        }
29
30        //Feltölti az attribútumokat
31        for(int i = 0; i < authorNodes.item(item).getAttributes().getLength(); i++){
32            String nodeAttributeName = authorNodes.item(item).getAttributes().item(i).getNodeName();
33            String nodeAttributeContent = authorNodes.item(item).getAttributes().item(i).getTextContent();
34
35            if(nodeAttributeName.equals("AuthorID")){author.authorID = Integer.parseInt(nodeAttributeContent);}
36        }
37
38        return author;
39    }
40
41    //Készít egy Author osztály alapú listát az XML dokumentumból kapott Author Node listájából
42    public static List<Author> makeAuthorList(NodeList authorNodeList){
43        List<Author> authorList = new ArrayList<Author>();
44        for(int i = 0; i < authorNodeList.getLength(); i++){
45            authorList.add(authorLoad(authorNodeList, i));
46        }
47        return authorList;
48    }
49 }
50
```

```

1 package hu.domparsa.a7tjix.elements;
2
3 import org.w3c.dom.NodeList;
4
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class Book {
9     //Attribútumok
10    public int bookID;
11    public int bSeries;
12
13    //Elemek
14    public String bookName;
15    public String bookPrice;
16    public int numOfPages;
17    public int numOfChapters;
18    public int volumeNumber;
19
20
21    //Book példány feltöltésére szolgál (segédfüggvény)
22    public static Book bookLoad(NodeList bookNodes, int item){
23        Book book = new Book();
24
25        //Feltölti az elemeket
26        for(int i = 0; i < bookNodes.item(item).getChildNodes().getLength(); i++){
27            String nodeName = bookNodes.item(item).getChildNodes().item(i).getNodeName();
28            String nodeContent = bookNodes.item(item).getChildNodes().item(i).getTextContent();
29            if(nodeName.equals("BookName")){book.bookName = nodeContent;}
30            else if(nodeName.equals("BookPrice")){book.bookPrice = nodeContent;}
31            else if(nodeName.equals("NumOfPages")){book.numOfPages = Integer.parseInt(nodeContent);}
32            else if(nodeName.equals("NumOfChapters")){book.numOfChapters = Integer.parseInt(nodeContent);}
33            else if(nodeName.equals("VolumeNumber")){book.volumeNumber = Integer.parseInt(nodeContent);}
34        }
35
36        //Feltölti az attribútumokat
37        for(int i = 0; i < bookNodes.item(item).getAttributes().getLength(); i++){
38            String nodeAttributeName = bookNodes.item(item).getAttributes().item(i).getNodeName();
39            String nodeAttributeContent = bookNodes.item(item).getAttributes().item(i).getTextContent();
40
41            if(nodeAttributeName.equals("BookID")){book.bookID = Integer.parseInt(nodeAttributeContent);}
42            else if(nodeAttributeName.equals("BSeries")){book.bSeries = Integer.parseInt(nodeAttributeContent);}
43        }
44
45        return book;
46    }
47
48    //Book egy Book osztály alapú listát az XML dokumentumból kapott Book Node listájából
49    public static List<Book> makeBookList(NodeList bookNodeList){
50        List<Book> bookList = new ArrayList<Book>();
51        for(int i = 0; i < bookNodeList.getLength(); i++){
52            bookList.add(bookLoad(bookNodeList, i));
53        }
54        return bookList;
55    }

```

```

1 package hu.domparsa.a7tjx.elements;
2
3 import org.w3c.dom.NodeList;
4
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class Bookstore {
9     //Attribútumok
10    public int bookstoreID;
11
12    //Elemek
13    public String bookstoreName;
14    public String sCity;
15    public String sStreet;
16    public String sHouseNumber;
17
18    //Bookstore példány feltöltésére szolgál (segédfüggvény)
19    public static Bookstore bookstoreLoad(NodeList bookstoreNodes, int item){
20        Bookstore bookstore = new Bookstore();
21
22        //Feltölti az elemeket
23        for(int i = 0; i < bookstoreNodes.item(item).getChildNodes().getLength(); i++){
24            String nodeName = bookstoreNodes.item(item).getChildNodes().item(i).getNodeName();
25            String nodeContent = bookstoreNodes.item(item).getChildNodes().item(i).getTextContent();
26            if(nodeName.equals("BookstoreName")){bookstore.bookstoreName = nodeContent;}
27            else if(nodeName.equals("StoreAddress")){
28                for(int j = 0; j < bookstoreNodes.item(item).getChildNodes().item(i).getChildNodes().getLength(); j++){
29                    String addressName = bookstoreNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getNodeName();
30                    String addressContent = bookstoreNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getTextContent();
31                    if(addressName.equals("SCity")){bookstore.sCity = addressContent;}
32                    else if(addressName.equals("SStreet")){bookstore.sStreet = addressContent;}
33                    else if(addressName.equals("SHouseNumber")){bookstore.sHouseNumber = addressContent;}
34                }
35            }
36        }
37
38        //Feltölti az attribútumokat
39        for(int i = 0; i < bookstoreNodes.item(item).getAttributes().getLength(); i++){
40            String nodeAttributeName = bookstoreNodes.item(item).getAttributes().item(i).getNodeName();
41            String nodeAttributeContent = bookstoreNodes.item(item).getAttributes().item(i).getTextContent();
42
43            if(nodeAttributeName.equals("BookstoreID")){bookstore.bookstoreID = Integer.parseInt(nodeAttributeContent);}
44        }
45
46        return bookstore;
47    }
48
49    //Készít egy Bookstore osztály alapú listát az XML dokumentumból kapott Bookstore Node listájából
50    public static List<Bookstore> makeBookstoreList(NodeList bookstoreNodeList){
51        List<Bookstore> bookstoreList = new ArrayList<Bookstore>();
52        for(int i = 0; i < bookstoreNodeList.getLength(); i++){
53            bookstoreList.add(bookstoreLoad(bookstoreNodeList, i));
54        }
55        return bookstoreList;

```

```

1 package hu.domparsa.a7tjix.elements;
2
3 import org.w3c.dom.NodeList;
4
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class Magazine {
9     //Attribútumok
10    public int magazineID;
11    public int mPublisher;
12
13    //Elemek
14    public String magazineName;
15    public String publishingFrequency;
16    public String demographic;
17    public String magazinePrice;
18
19    //Magazine példány feltöltésére szolgál (segédfüggvény)
20    public static Magazine magazineLoad(NodeList magazineNodes, int item){
21        Magazine magazine = new Magazine();
22
23        //Feltölti az elemeket
24        for(int i = 0; i < magazineNodes.item(item).getChildNodes().getLength(); i++){
25            String nodeName = magazineNodes.item(item).getChildNodes().item(i).getNodeName();
26            String nodeContent = magazineNodes.item(item).getChildNodes().item(i).getTextContent();
27            if(nodeName.equals("MagazineName")){magazine.magazineName = nodeContent;}
28            else if(nodeName.equals("PublishingFrequency")){magazine.publishingFrequency = nodeContent;}
29            else if(nodeName.equals("Demographic")){magazine.demographic = nodeContent;}
30            else if(nodeName.equals("MagazinePrice")){magazine.magazinePrice = nodeContent;}
31        }
32
33        //Feltölti az attribútumokat
34        for(int i = 0; i < magazineNodes.item(item).getAttributes().getLength(); i++){
35            String nodeAttributeName = magazineNodes.item(item).getAttributes().item(i).getNodeName();
36            String nodeAttributeContent = magazineNodes.item(item).getAttributes().item(i).getTextContent();
37
38            if(nodeAttributeName.equals("MagazineID")){magazine.magazineID = Integer.parseInt(nodeAttributeContent);}
39            else if(nodeAttributeName.equals("MPublisher")){magazine.mPublisher = Integer.parseInt(nodeAttributeContent);}
40        }
41
42        return magazine;
43    }
44
45    //Készít egy Magazine osztály alapú listát az XML dokumentumból kapott Magazine Node listájából
46    public static List<Magazine> makeMagazineList(NodeList magazineNodeList){
47        List<Magazine> magazineList = new ArrayList<Magazine>();
48        for(int i = 0; i < magazineNodeList.getLength(); i++){
49            magazineList.add(magazineLoad(magazineNodeList, i));
50        }
51        return magazineList;
52    }
53 }
54
55

```

```

1 package hu.domparse.a7tjix.elements;
2
3 import org.w3c.dom.NodeList;
4
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class President {
9     //Attribútumok
10    public int presidentID;
11
12    //Elemek
13    public String presidentName;
14    public String presidentGender;
15    public int presidentAge;
16
17    //President példány feltöltésére szolgál (segédfüggvény)
18    public static President presidentLoad(NodeList presidentNodes, int item){
19        President president = new President();
20
21        //Feltölti az elemeket
22        for(int i = 0; i < presidentNodes.item(item).getChildNodes().getLength(); i++){
23            String nodeName = presidentNodes.item(item).getChildNodes().item(i).getNodeName();
24            String nodeContent = presidentNodes.item(item).getChildNodes().item(i).getTextContent();
25            if(nodeName.equals("PresidentName")){president.presidentName = nodeContent;}
26            else if(nodeName.equals("PresidentGender")){president.presidentGender = nodeContent;}
27            else if(nodeName.equals("PresidentAge")){president.presidentAge = Integer.parseInt(nodeContent);}
28        }
29
30        //Feltölti az attribútumokat
31        for(int i = 0; i < presidentNodes.item(item).getAttributes().getLength(); i++){
32            String nodeAttributeName = presidentNodes.item(item).getAttributes().item(i).getNodeName();
33            String nodeAttributeContent = presidentNodes.item(item).getAttributes().item(i).getTextContent();
34
35            if(nodeAttributeName.equals("PresidentID")){president.presidentID = Integer.parseInt(nodeAttributeContent);}
36        }
37
38        return president;
39    }
40
41    //Készít egy President osztály alapú listát az XML dokumentumból kapott President Node listájából
42    public static List<President> makePresidentList(NodeList presidentNodeList){
43        List<President> presidentList = new ArrayList<President>();
44        for(int i = 0; i < presidentNodeList.getLength(); i++){
45            presidentList.add(presidentLoad(presidentNodeList, i));
46        }
47        return presidentList;
48    }
49 }
50

```



```

1 package hu.domparse.a7tjx.elements;
2
3 import org.w3c.dom.NodeList;
4
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class Publisher {
9     //Attribútumok
10    public int publisherID;
11    public int pPresident;
12
13    //Elemek
14    public String publisherName;
15    public String phoneNumber;
16    public String emailAddress;
17    public String pCity;
18    public String pStreet;
19    public String pHouseNumber;
20
21    //Publisher példány feltöltésére szolgál (segédfüggvény)
22    public static Publisher publisherLoad(NodeList publisherNodes, int item){
23        Publisher publisher = new Publisher();
24
25        //Feltölti az elemeket
26        for(int i = 0; i < publisherNodes.item(item).getChildNodes().getLength(); i++){
27            String nodeName = publisherNodes.item(item).getChildNodes().item(i).getNodeName();
28            String nodeContent = publisherNodes.item(item).getChildNodes().item(i).getTextContent();
29            if(nodeName.equals("PublisherName")){publisher.publisherName = nodeContent;}
30            else if(nodeName.equals("PhoneNumber")){publisher.phoneNumber = nodeContent;}
31            else if(nodeName.equals("EmailAddress")){publisher.emailAddress = nodeContent;}
32            else if(nodeName.equals("HQAddress")){
33                for(int j = 0; j < publisherNodes.item(item).getChildNodes().item(i).getChildNodes().getLength(); j++){
34                    String addressName = publisherNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getNodeName();
35                    String addressContent = publisherNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getTextContent();
36                    if(addressName.equals("PCity")){publisher.pCity = addressContent;}
37                    else if(addressName.equals("PStreet")){publisher.pStreet = addressContent;}
38                    else if(addressName.equals("PHouseNumber")){publisher.pHouseNumber = addressContent;}
39                }
40            }
41        }
42
43        //Feltölti az attribútumokat
44        for(int i = 0; i < publisherNodes.item(item).getAttributes().getLength(); i++){
45            String nodeAttributeName = publisherNodes.item(item).getAttributes().item(i).getNodeName();
46            String nodeAttributeContent = publisherNodes.item(item).getAttributes().item(i).getTextContent();
47
48            if(nodeAttributeName.equals("PublisherID")){publisher.publisherID = Integer.parseInt(nodeAttributeContent);}
49            else if(nodeAttributeName.equals("PPresident")){publisher.pPresident = Integer.parseInt(nodeAttributeContent);}
50        }
51
52        return publisher;
53    }
54
55    //Készít egy Publisher osztály alapú listát az XML dokumentumból kapott Publisher Node listájából
56    public static List<Publisher> makePublisherList(NodeList publisherNodeList){
57        List<Publisher> publisherList = new ArrayList<Publisher>();
58        for(int i = 0; i < publisherNodeList.getLength(); i++){
59            publisherList.add(publisherLoad(publisherNodeList, i));
60        }
61        return publisherList;
62    }
63 }
64

```

```

DOMParserA7Tijx / DOMParserA7Tijx / hu / domparse / a7tjx / elements / Selling.java
1 package hu.domparse.a7tjx.elements;
2
3 import org.w3c.dom.NodeList;
4
5 import java.util.ArrayList;
6 import java.util.List;
7
8 public class Selling {
9     //Attribútumok
10    public int sBookID;
11    public int sBookstoreID;
12
13    //Elemek
14    public int numOfSells;
15
16    //Selling példány feltöltésére szolgál (segédfüggvény)
17    public static Selling sellingLoad(NodeList sellingNodes, int item){
18        Selling selling = new Selling();
19
20        //Feltölti az elemeket
21        for(int i = 0; i < sellingNodes.item(item).getChildNodes().getLength(); i++){
22            String nodeName = sellingNodes.item(item).getChildNodes().item(i).getNodeName();
23            String nodeContent = sellingNodes.item(item).getChildNodes().item(i).getTextContent();
24
25            if(nodeName.equals("NumOfSells")){selling.numOfSells = Integer.parseInt(nodeContent);}
26        }
27
28        //Feltölti az attribútumokat
29        for(int i = 0; i < sellingNodes.item(item).getAttributes().getLength(); i++){
30            String nodeAttributeName = sellingNodes.item(item).getAttributes().item(i).getNodeName();
31            String nodeAttributeContent = sellingNodes.item(item).getAttributes().item(i).getTextContent();
32
33            if(nodeAttributeName.equals("SBookID")){selling.sBookID = Integer.parseInt(nodeAttributeContent);}
34            else if(nodeAttributeName.equals("SBookstoreID")){selling.sBookstoreID = Integer.parseInt(nodeAttributeContent);}
35        }
36
37        return selling;
38    }
39
40    //Készít egy Selling osztály alapú listát az XML dokumentumból kapott Selling Node listájából
41    public static List<Selling> makeSellingList(NodeList sellingNodeList){
42        List<Selling> sellingList = new ArrayList<Selling>();
43        for(int i = 0; i < sellingNodeList.getLength(); i++){
44            sellingList.add(sellingLoad(sellingNodeList, i));
45        }
46        return sellingList;
47    }
48 }
49

```

```

1 package hu.domparse.a7tjix.elements;
2
3 import org.w3c.dom.Node;
4 import org.w3c.dom.NodeList;
5
6 import java.util.ArrayList;
7 import java.util.List;
8
9 public class Series{
10     //Attribútumok
11     public int seriesID;
12     public int sMagazine;
13     public int sAuthor;
14
15     //Elemek
16     public String seriesName;
17     public String ranking;
18     public int numOfReaders;
19     public List<String> subGenre;
20
21     //Series példány feltöltésére szolgál (segédfüggvény)
22     public static Series seriesLoad(NodeList seriesNodes, int item){
23         Series series = new Series();
24
25         //Feltölti az elemeket
26         for(int i = 0; i < seriesNodes.item(item).getChildNodes().getLength(); i++){
27             String nodeName = seriesNodes.item(item).getChildNodes().item(i).getNodeName();
28             String nodeContent = seriesNodes.item(item).getChildNodes().item(i).getTextContent();
29             if(nodeName.equals("SeriesName")){series.seriesName = nodeContent;}
30             else if(nodeName.equals("Ranking")){series.ranking = nodeContent;}
31             else if(nodeName.equals("NumOfReaders")){series.numOfReaders = Integer.parseInt(nodeContent);}
32             else if(nodeName.equals("Genre")){
33                 List<String> subgenreList = new ArrayList<String>();
34                 for (int j = 0; j < seriesNodes.item(item).getChildNodes().item(i).getChildNodes().getLength(); j++ ) {
35                     if(seriesNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getNodeType() == Node.ELEMENT_NODE){
36                         subgenreList.add(seriesNodes.item(item).getChildNodes().item(i).getChildNodes().item(j).getTextContent());
37                     }
38                 }
39                 series.subGenre = subgenreList;
40             }
41         }
42     }
43
44     //Feltölti az attribútumokat
45     for(int i = 0; i < seriesNodes.item(item).getAttributes().getLength(); i++){
46         String nodeAttributeName = seriesNodes.item(item).getAttributes().item(i).getNodeName();
47         String nodeAttributeContent = seriesNodes.item(item).getAttributes().item(i).getTextContent();
48
49         if(nodeAttributeName.equals("SeriesID")){series.seriesID = Integer.parseInt(nodeAttributeContent);}
50         else if(nodeAttributeName.equals("SMagazine")){series.sMagazine = Integer.parseInt(nodeAttributeContent);}
51         else if(nodeAttributeName.equals("SAuthor")){series.sAuthor = Integer.parseInt(nodeAttributeContent);}
52     }
53
54     return series;
55 }
56

```

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=Shueisha, PhoneNumber=090-1731-1447, EmailAddress=doe@shueisha.co.jp, HQAddress: PCity=Tokyo, PStreet=Hitotsubashi Chiyoda-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 FileOutputStream 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) ---

```

1 package hu.domparsing.a7tjix;
2
3 import hu.domparsing.a7tjix.elements.*;
4 import org.w3c.dom.Document;
5 import org.w3c.dom.NodeList;
6
7 import javax.xml.parsers.DocumentBuilder;
8 import javax.xml.parsers.DocumentBuilderFactory;
9 import javax.xml.transform.Transformer;
10 import javax.xml.transform.TransformerFactory;
11 import javax.xml.transform.dom.DOMSource;
12 import javax.xml.transform.stream.StreamResult;
13 import java.io.FileOutputStream;
14 import java.util.List;
15
16 import static hu.domparsing.a7tjix.elements.Author.makeAuthorList;
17 import static hu.domparsing.a7tjix.elements.Book.makeBookList;
18 import static hu.domparsing.a7tjix.elements.Bookstore.makeBookstoreList;
19 import static hu.domparsing.a7tjix.elements.Magazine.makeMagazineList;
20 import static hu.domparsing.a7tjix.elements.President.makePresidentList;
21 import static hu.domparsing.a7tjix.elements.Publisher.makePublisherList;
22 import static hu.domparsing.a7tjix.elements.Selling.makeSellingList;
23 import static hu.domparsing.a7tjix.elements.Series.makeSeriesList;
24
25 public class DomModifyA7TIJX {
26     public static void main(String[] args) {
27         try{
28             //XML dokumentum read
29             DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
30             DocumentBuilder builder = builderFactory.newDocumentBuilder();
31             Document document = builder.parse("XMLA7TIJX.xml");
32             document.getDocumentElement().normalize();
33
34             //XML dokumentum modify
35             modifyElementXML(document, "Series", 0, "SeriesName", "Bomby Girl");
36             modifyElementXML(document, "Magazine", 2, "PublishingFrequency", "Monthly");
37             modifyElementXML(document, "Publisher", 1, "PhoneNumber", "123-4567-8910");
38             modifyElementXML(document, "President", 0, "PresidentGender", "Female");
39             modifyAttributeXML(document, "Selling", 4, "SBookID", "55");
40             modifyAttributeXML(document, "Book", 3, "BSeries", "4");
41

```

```

42         } catch (Exception e) {
43             //XML dokumentum write
44             TransformerFactory transformerFactory = TransformerFactory.newInstance();
45             Transformer transformer = transformerFactory.newTransformer();
46             DOMSource domSource = new DOMSource(document);
47             FileOutputStream output = new FileOutputStream("XMLA7TIJXmodify.xml");
48             StreamResult streamResult = new StreamResult(output);
49             transformer.transform(domSource, streamResult);
50
51         } catch (Exception ee) {
52             e.printStackTrace();
53         }
54     }
55
56     catch (Exception ee) {
57         ee.printStackTrace();
58     }
59 }
60
61 //Ha elemet akarunk módosítani
62 public static void modifyElementXML(Document document, String mainTag, int mainTagItem, String targetTag, String modifiedString){
63     for(int i = 0; i < document.getElementsByTagName(mainTag).item(mainTagItem).getChildNodes().getLength(); i++){
64         if(document.getElementsByTagName(mainTag).item(mainTagItem).getChildNodes().item(i).getNodeName() == targetTag){
65             System.out.println("Before: " + document.getElementsByTagName(mainTag).item(mainTagItem).getChildNodes().item(i).getTextContent());
66             document.getElementsByTagName(mainTag).item(mainTagItem).getChildNodes().item(i).setTextContent(modifiedString);
67             System.out.println("After: " + document.getElementsByTagName(mainTag).item(mainTagItem).getChildNodes().item(i).getTextContent());
68         }
69     }
70     System.out.println("-----");
71 }
72
73 //Ha attribútumot akarunk módosítani
74 public static void modifyAttributeXML(Document document, String mainTag, int mainTagItem, String targetAttribute, String modifiedString){
75     for(int i = 0; i < document.getElementsByTagName(mainTag).item(mainTagItem).getAttributes().getLength(); i++){
76         if(document.getElementsByTagName(mainTag).item(mainTagItem).getAttributes().item(i).getNodeName() == targetAttribute){
77             System.out.println("Before: " + document.getElementsByTagName(mainTag).item(mainTagItem).getAttributes().item(i).getTextContent());
78             document.getElementsByTagName(mainTag).item(mainTagItem).getAttributes().item(i).setTextContent(modifiedString);
79             System.out.println("After: " + document.getElementsByTagName(mainTag).item(mainTagItem).getAttributes().item(i).getTextContent());
80         }
81     }
82     System.out.println("-----");
83 }
84
85 }
86
87 }

```


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>75000</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>
  <HQAAddress>
    <PCity>Tokyo</PCity>
    <PStreet>Hitotsubashi Chiyoda-ku</PStreet>
    <PHouseNumber>2-3-1</PHouseNumber>
  </HQAAddress>
</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) ---

```
1 package hu.domparsing.a7tjix;
2
3 import hu.domparsing.a7tjix.elements.*;
4 import org.w3c.dom.Document;
5 import org.w3c.dom.NodeList;
6
7 import javax.xml.parsers.DocumentBuilder;
8 import javax.xml.parsers.DocumentBuilderFactory;
9 import java.util.ArrayList;
10 import java.util.List;
11
12 import static hu.domparsing.a7tjix.elements.Author.makeAuthorList;
13 import static hu.domparsing.a7tjix.elements.Book.makeBookList;
14 import static hu.domparsing.a7tjix.elements.Bookstore.makeBookstoreList;
15 import static hu.domparsing.a7tjix.elements.Magazine.makeMagazineList;
16 import static hu.domparsing.a7tjix.elements.Publisher.makePublisherList;
17 import static hu.domparsing.a7tjix.elements.Selling.makeSellingList;
18 import static hu.domparsing.a7tjix.elements.Series.makeSeriesList;
19
20
21 public class DomQueryA7TIJX {
22     public static void main(String[] args) {
23         try {
24             //XML dokumentum read
25             DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
26             DocumentBuilder builder = builderFactory.newDocumentBuilder();
27             Document document = builder.parse("XMLA7TIJX.xml");
28             document.getDocumentElement().normalize();
29
30             //Node listák kiszedése a dokumentumból
31             NodeList kkiado_a7tjix = document.getElementsByTagName("Kepregeny_kiado_A7TIJX");
32
33             NodeList SeriesNodeList = document.getElementsByTagName("Series");
34             NodeList AuthorNodeList = document.getElementsByTagName("Author");
35             NodeList MagazineNodeList = document.getElementsByTagName("Magazine");
36             NodeList PublisherNodeList = document.getElementsByTagName("Publisher");
37             NodeList PresidentNodeList = document.getElementsByTagName("President");
38             NodeList BookNodeList = document.getElementsByTagName("Book");
39             NodeList SellingNodeList = document.getElementsByTagName("Selling");
40             NodeList BookstoreNodeList = document.getElementsByTagName("Bookstore");
41
42             //Node listákból elem listákat csinálunk
43             List<Series> seriesList = makeSeriesList(SeriesNodeList);
44             List<Author> authorList = makeAuthorList(AuthorNodeList);
45             List<Magazine> magazineList = makeMagazineList(MagazineNodeList);
46             List<Publisher> publisherList = makePublisherList(PublisherNodeList);
47             List<President> presidentList = makePresidentList(PresidentNodeList);
48             List<Book> bookList = makeBookList(BookNodeList);
49             List<Selling> sellingList = makeSellingList(SellingNodeList);
50             List<Bookstore> bookstoreList = makeBookstoreList(BookstoreNodeList);
51         }
52     }
53 }
```

```

52 //Lekérdezések
53
54 //Attack on Titan képregény rankja
55 System.out.println("Attack on Titan képregény rankja: ");
56 for (Series s : seriesList) {
57     if(s.seriesName.equals("Attack on Titan")){
58         System.out.println(s.ranking);
59     }
60 }
61 System.out.println("-----");
62
63 //A Sci-Fi műfajban jártas írók neve
64 System.out.println("A Sci-Fi műfajban jártas írók neve: ");
65 List<Integer> iID = new ArrayList<Integer>();
66 for (Series s : seriesList) {
67     if(s.subGenre.contains("Sci-fi")) {
68         iID.add(s.sAuthor);
69     }
70 }
71 for (Author a : authorList) {
72     for (int i : iID) {
73         if(a.authorID == i){
74             System.out.println(a.authorName);
75         }
76     }
77 }
78 System.out.println("-----");
79
80 //Book First Umeda könyvesboltban árult könyvek
81 System.out.println("Book First Umeda könyvesboltban árult könyvek: ");
82 int bsID = -1;
83 List<Integer> bukID = new ArrayList<Integer>();
84 for (Bookstore bs : bookstoreList) {
85     if(bs.bookstoreName.equals("Book First Umeda")){
86         bsID = bs.bookstoreID;
87     }
88 }
89 for (Selling sell : sellingList) {
90     if(sell.sBookstoreID == bsID){
91         bukID.add(sell.sBookID);
92     }
93 }
94 for (Book b : bookList) {
95     for (int i : bukID) {
96         if(b.bookID == i){
97             System.out.println(b.bookName);
98         }
99     }
100 }
101 System.out.println("-----");

```

```

103 //Urasawa Naoki kiadójának a teljes címe
104 System.out.println("Urasawa Naoki kiadójának a teljes címe: ");
105 int uID = -1, mID = -1;
106 for (Author auth : authorList) {
107     if(auth.authorName.equals("Urasawa Naoki")){
108         uID = auth.authorID;
109     }
110 }
111 for (Series si : seriesList) {
112     if(si.sAuthor == uID){
113         mID = si.sMagazine;
114     }
115 }
116 for (Magazine m : magazineList) {
117     if(m.magazineID == mID){
118         for (Publisher p : publisherList){
119             if(m.mPublisher == p.publisherID){
120                 System.out.println(p.publisherName);
121                 System.out.println(p.pCity);
122                 System.out.println(p.pStreet);
123                 System.out.println(p.pHouseNumber);
124             }
125         }
126     }
127 }
128 System.out.println("-----");
129
130 //Azoknak a könyveknek a nevének a lekérdezése amelyeket Fujimoto Tatsuki írt
131 int aID = -1;
132 List<Integer> sID = new ArrayList<Integer>();
133 for (Author auth : authorList) {
134     if(auth.authorName.equals("Fujimoto Tatsuki")){
135         aID = auth.authorID;
136     }
137 }
138 for (Series seri : seriesList) {
139     if(seri.sAuthor == aID){
140         sID.add(seri.seriesID);
141     }
142 }
143 System.out.println("Fujimoto Tatsuki által készített könyvek:");
144 for(Book book : bookList) {
145     for (int s : sID) {
146         if(book.bSeries == s){
147             System.out.println(book.bookName);
148         }
149     }
150 }
151 System.out.println("-----");
152 }
153 catch (Exception ee) {
154     ee.printStackTrace();
155 }

```

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, akár csak a modify esetében. Betápláltam a dokumentumot majd az felhasználva létrehoztam egy új fájlt.

(DomWriteA7TIJX.java) ---

```
1 package hu.domparse.a7tijx;
2
3 import org.w3c.dom.Document;
4
5 import javax.xml.parsers.DocumentBuilder;
6 import javax.xml.parsers.DocumentBuilderFactory;
7 import javax.xml.transform.Transformer;
8 import javax.xml.transform.TransformerFactory;
9 import javax.xml.transform.dom.DOMSource;
10 import javax.xml.transform.stream.StreamResult;
11 import java.io.FileOutputStream;
12
13 public class DOMWriteA7TIJX {
14     public static void main(String[] args) {
15         try{
16             //XML dokumentum read
17             DocumentBuilderFactory builderFactory = DocumentBuilderFactory.newInstance();
18             DocumentBuilder builder = builderFactory.newDocumentBuilder();
19             Document document = builder.parse("XMLA7TIJX.xml");
20             document.getDocumentElement().normalize();
21
22             try {
23                 //XML dokumentum write
24                 TransformerFactory transformerFactory = TransformerFactory.newInstance();
25                 Transformer transformer = transformerFactory.newTransformer();
26                 DOMSource domSource = new DOMSource(document);
27                 FileOutputStream output = new FileOutputStream("XMLA7TIJX1.xml");
28                 StreamResult streamResult = new StreamResult(output);
29                 transformer.transform(domSource, streamResult);
30                 transformer.transform(domSource, new StreamResult(System.out));
31             } catch (Exception e) {
32                 e.printStackTrace();
33             }
34         };
35     }
36 }
37 catch (Exception ee) {
38     ee.printStackTrace();
39 }
40
41 }
42 }
43
```