JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Könyvtár rendszer

Készítette: **Holczer Vanda**

Neptunkód: **C0LLER**

Dátum: 2023.12.04.

Tartalomjegyzék

Bevezetés

A nagyobb könyvtáraknak sok esetben jelentős mennyiségű adatot kell tárolniuk a kölcsönzésekről és a könyvekről, ehhez nagy segítséget nyújthat egy elektronikus adatbázis, a következőkben egy ilyen modellről lesz szó.

A feladat leírása

Az első egyed a Business card, ami a Névjegykártya. Rendelkezik egy Card ID kulcstulajdonsággal, valamint egy Written name tulajdonsággal, ami a ráírt név.

A második egyed a Librarian, vagyis a könyvtáros. Ez az egyed a következő tulajdonságokkal rendelkezik: Librarian ID, ez a kulcstulajdonság, LName, a könyvtáros neve, ez egy összetett tulajdonság, a vezeték (LLastname)- és a keresztnévből (LFirst name) áll, LDate of birth, a születési dátum, LAge, kor, ami egy származtatott tulajdonság, LPhone number, telefonszám, ez pedig egy többértékű tulajdonság.

A következő egyedünk a Borrower, ez az a személy aki kikölcsönzi a könyvet. Neki szintén van egy azonosítója, amely a kulcstulajdonság, a Borrower ID, neve is ami a BName, ez itt is összetett tulajdonság, a BFirst name (keresztnév)-ből és a BLast name (vezetéknév)-ből épül fel, valamint születési dátuma (BDate of birth). A lakcím (BAdress) egy összetett tulajdonság, a városból (City), utcából (Street) és házszámból (House number) áll.Társul ehhez az egyedhez egy Email address és egy BPhone number tulajdonság is, melyek közül mindkettő opcionális, a telefonszám ezenfelül többértékű is.

A harmadik egyed a Book, azaz a könyv. Kulcstulajdonsága a Book ID, ami egy azonosító és értelemszerűen egyedi, ezenkívül van címe (Title) és egy Borrowable tulajdonsága, ez a kölcsönözhetőséget jelöli.

Az utolsó egyed az Author, a könyv szerzője. Egyedi, kulcstulajdonsága az Author ID, rendelkezik névvel (AName), ami összetett tulajdonság az AFirst name (keresztnév)-ből és a ALast name (vezetéknév)-ből épül fel, tárolva van a születési dátuma (ADate of birth) és az ebből következő származtatott tulajdonság, AAge (kor).

Az adott egyedek különböző kapcsolatban állnak egymással, melyek a következők:

A Business card és a Librarian között az Owned by kapcsolat áll, amely a birtoklást jelenti, ez egy kötelező egy-egy kapcsolat, aminek nincsen tulajdonsága.

A Librarian és a Borrower között a Handing over kapcsolat van, ennek van egy Date tulajdonsága, ami a kölcsönzés dátuma, egy Book ID, ami a könyvvel köti össze és egy Lending/Taking back, ami azt jelöli, hogy kölcsönadták vagy visszahozták a könyvet. Ez egy kötelező, több-több kapcsolat.

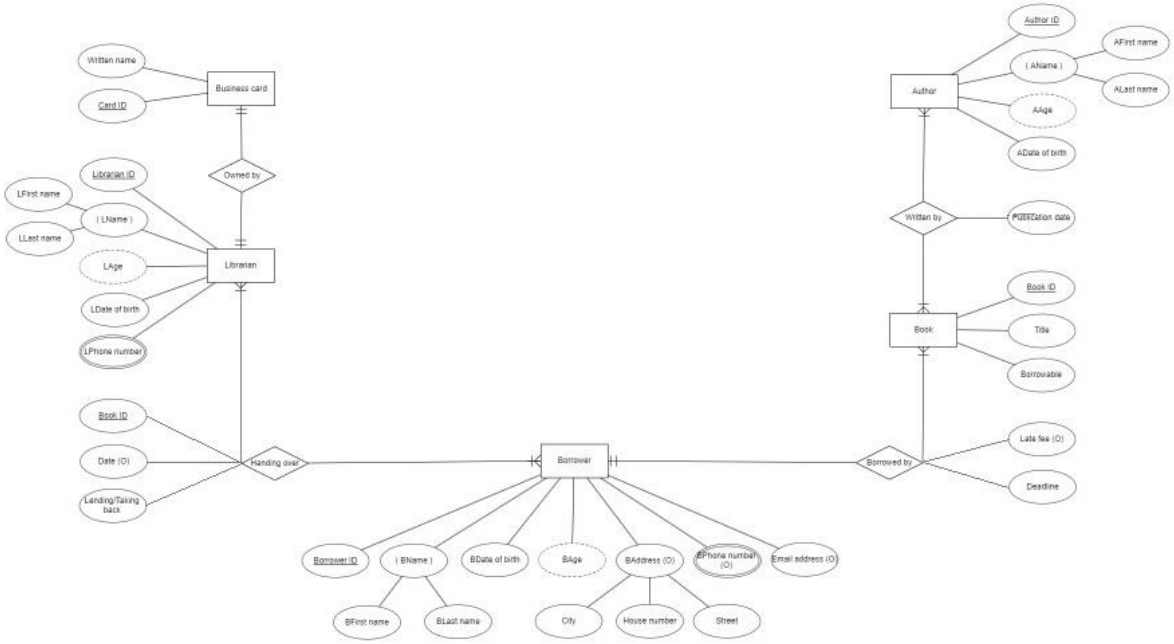
A Book és az Author kötelező több-több kapcsolatban áll egymással (Written by). A kapcsolatnak van egy tulajdonsága ami a kiadás dátumát jelöli (Publication date).

Továbbá a Book kötelező, egy-több kapcsolatban áll a Borrowerrel. Tulajdonságai a Late fee, vagyis a késedelmi díj, opcionális mert csak akkor van, ha nem vitték vissza időben a könyvet, valamint a Deadline, ami a kölcsönzési határidő.

1. feladat

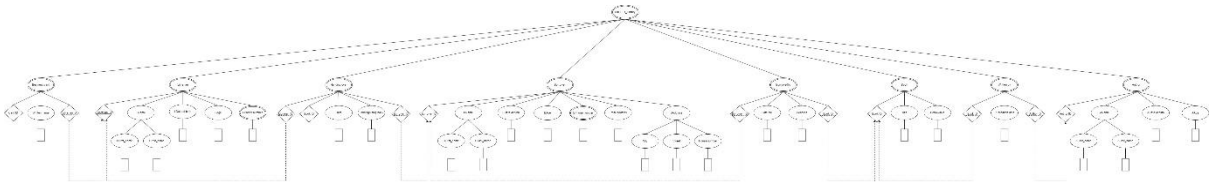
1a) Az adatbázis ER modell tervezése

Az ER-modell elkészítése során az ERDPlus-t használtam. Első lépésként az egyedeket különböző hoztam létre, amleyeket téglalapok jelölnek, majd a hozzájuk tartozó tulajdonságokat elipszisekkel jelenítettem meg. Jelöltem a többértékű tulajdonságokat (dupla körvonallal), valamint az opcionális tulajdonságokat. Ezután az egyedek közötti kapcsolatokat hoztam létre, amelyeket rombuszok reprezentálnak. A kapcsolatoknál jelöltem a kötelezőséget, valamint, hogy milyen típusú kapcsolatról van szó (1:N, N:M, 1:1), majd hozzáadtam a tulajdonságaikat.



1b) Az adatbázis konvertálása XDM modellre

A gyökérelemet egy elipszissel ábrázoltam, ebből jönnek le az egyedek, amelyeket szintén ellipszis jelöl, ahogy az egyedek tulajdonságait is. A tulajdonságokból téglalapok jönnek ki. A kulcstulajdonságokat rombusz jelöli, valamint szagatott vonalas nyíl köti össze őket, amely az idegenkulcsból indul, ez jelképezi a kapcsolatokat.



1c) Az XDM modell alapján XML dokumentum készítése

Az XDM modell segítségével megírtam az XML fájlt. A fájl elején a Prolog található, ezt követően a gyökérelem. Utána az egyes egyedek, valamint a kapcsolótáblák példányosítása következik, először létrehoztam a vázukat, majd feltöltöttem őket adatokkal.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<C0LLER_Library xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:noNamespaceSchemaLocation="XMLSchemaC0LLER.xsd">

    <!-- Librarian instances -->

    <librarian librarian_id="01">
        <name>
            <first_name>Klaudia</first_name>
            <last_name>Fehér</last_name>
        </name>
        <date_of_birth>2000-05-17</date_of_birth>
        <age>23</age>
        <phone_number>0630-080-8126</phone_number>
        <phone_number>0620-871-4015</phone_number>
    </librarian>

    <librarian librarian_id="02">
        <name>
            <first_name>Albert</first_name>
            <last_name>Pintér</last_name>
        </name>
        <date_of_birth>1988-12-17</date_of_birth>
        <age>35</age>
        <phone_number>0620-155-0102</phone_number>
    </librarian>

    <librarian librarian_id="03">
        <name>
            <first_name>Erzsébet</first_name>
            <last_name>Farkas</last_name>
        </name>
        <date_of_birth>1966-03-12</date_of_birth>
        <age>57</age>
        <phone_number>0670-440-8020</phone_number>
    </librarian>


    <!-- Business card instances -->
```

```xml
<business_card card_id="11" librarian_id="01">
    <written_name>Fehér Klaudia</written_name>
</business_card>

<business_card card_id="12" librarian_id="02">
    <written_name>Pintér Albert</written_name>
</business_card>

<business_card card_id="13" librarian_id="03">
    <written_name>Farkas Erzsébet</written_name>
</business_card>

<!-- Borrower instances -->

<borrower borrower_id="21">
    <name>
        <first_name>Árpád</first_name>
        <last_name>Jakab</last_name>
    </name>
    <date_of_birth>1965-01-12</date_of_birth>
    <age>58</age>
    <address>
        <city>Miskolc</city>
        <street>Vitéz utca</street>
        <house_number>31</house_number>
    </address>
    <phone_number>0630-651-4518</phone_number>
    <phone_number>0670-749-3819</phone_number>
</borrower>

<borrower borrower_id="22">
    <name>
        <first_name>Georgina</first_name>
        <last_name>Somogyi</last_name>
    </name>
    <date_of_birth>1972-10-24</date_of_birth>
    <age>51</age>
    <address>
        <city>Arnót</city>
        <street>Kisfaludy utca</street>
        <house_number>17</house_number>
    </address>
    <phone_number>0620-845-4708</phone_number>
    <email_address>somogyigeorgina@citromail.hu</email_address>
</borrower>

<borrower borrower_id="23">
    <name>
```

```
            <first_name>Maja</first_name>
            <last_name>Rácz</last_name>
        </name>
        <date_of_birth>2000-11-28</date_of_birth>
        <age>23</age>
        <email_address>rmaja@freemail.hu</email_address>
    </borrower>

    <!-- Handing over switchboard instances -->

    <handing_over book_id="32" librarian_id="02" borrower_id="21">
        <lending_or_taking>1</lending_or_taking>
    </handing_over>

    <handing_over book_id="31" librarian_id="03" borrower_id="22">
        <handing_date>2023-08-23</handing_date>
        <lending_or_taking>1</lending_or_taking>
    </handing_over>

    <handing_over book_id="33" librarian_id="01" borrower_id="23">
        <handing_date>2022-12-16</handing_date>
        <lending_or_taking>0</lending_or_taking>
    </handing_over>

    <!-- Book instances -->

    <book book_id="31">
        <title> Harry Potter és a Bölcsek Köve</title>
        <borrowable>1</borrowable>
    </book>

    <book book_id="32">
        <title> Ahol a folyami rákok énekelnek</title>
        <borrowable>1</borrowable>
    </book>

    <book book_id="33">
        <title> Fordulópont</title>
        <borrowable>1</borrowable>
    </book>

    <!-- Borrowed by switchboard instances -->

    <borrowed_by book_id="32" borrower_id="21">
        <late_fee>0</late_fee>
        <deadline>2023-03-17</deadline>
    </borrowed_by>
```

```
    <borrowed_by book_id="31" borrower_id="22">
        <late_fee>0</late_fee>
        <deadline>2023-09-01</deadline>
    </borrowed_by>

    <borrowed_by book_id="33" borrower_id="23">
        <late_fee>800</late_fee>
        <deadline>2023-02-16</deadline>
    </borrowed_by>

    <!-- Author instances -->

    <author author_id="41">
        <name>
            <first_name>Joanne</first_name>
            <last_name>Rowling</last_name>
        </name>
        <date_of_birth>1965-07-31</date_of_birth>
        <age>58</age>
    </author>

    <author author_id="42">
        <name>
            <first_name>Delia</first_name>
            <last_name>Owens</last_name>
        </name>
        <date_of_birth>1949-04-04</date_of_birth>
        <age>74</age>
    </author>

    <author author_id="43">
        <name>
            <first_name>Danielle</first_name>
            <last_name>Steel</last_name>
        </name>
        <date_of_birth>1965-07-31</date_of_birth>
        <age>58</age>
    </author>

    <!-- Written by switchboard instances -->

    <written_by book_id="31" author_id="41">
        <publication_date>2001-12-13</publication_date>
    </written_by>

    <written_by book_id="32" author_id="42">
        <publication_date>2018-08-14</publication_date>
    </written_by>
```

```
    <written_by book_id="33" author_id="43">
        <publication_date>2019-07-21</publication_date>
    </written_by>

</C0LLER_Library>
```

1d) Az XML dokumentum alapján XMLSchema készítése

Az XML fájl alapján elkészítettem az XSD dokumentumot, vagyis az XMLSchema-t. Először definiáltam az egyszerű típusokat és a megkötéseket, ezt követően az összetett és saját típusokat. Utána felírtam az elemek felhasználását a gyökérelemtől, majd az egyedi- és idegenkulcsokat és végül az 1:1 kapcsolatot.

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


    <!-- Collection of simple types and restrections -->

    <xs:element name="date_of_birth" type="xs:date" />
    <xs:element name="age" type="xs:positiveInteger" />
    <xs:element name="phone_number" type="phoneNumberType" />
    <xs:element name="email_address" type="emailAddressType" />
    <xs:element name="written_name" type="xs:string" />
    <xs:element name="date" type="xs:date" />
    <xs:element name="lending_or_taking" type="xs:boolean" />
    <xs:element name="title" type="xs:string" />
    <xs:element name="borrowable" type="xs:boolean" />
    <xs:element name="late_fee" type="xs:integer" />
    <xs:element name="deadline" type="xs:date" />
    <xs:element name="publication_date" type="xs:date" />
    <xs:element name="handing_date" type="xs:date"></xs:element>

    <xs:simpleType name="phoneNumberType">
        <xs:restriction base="xs:string">
            <xs:pattern value="\d{4}-\d{3}-\d{4}" />
        </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="emailAddressType">
        <xs:restriction base="xs:string">
            <xs:pattern
                value="([0-9a-zA-Z]([-.\w]*[0-9a-zA-Z])*@([0-9a-zA-Z][-\w]*[0-
9a-zA-Z]\.)+[a-zA-Z]{2,9})" />
        </xs:restriction>
    </xs:simpleType>

    <!-- Complex type and own type definitions -->


    <xs:complexType name="addressType">
        <xs:sequence>
            <xs:element name="city" type="xs:string" />
```

```
            <xs:element name="street" type="xs:string" />
            <xs:element name="house_number" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="nameType">
        <xs:sequence>
            <xs:element name="first_name" type="xs:string" />
            <xs:element name="last_name" type="xs:string" />
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="librarianType">
        <xs:sequence>
            <xs:element name="name" type="nameType" />
            <xs:element ref="date_of_birth" />
            <xs:element ref="age" />
            <xs:element ref="phone_number" maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="librarian_id" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="businessCardType">
        <xs:sequence>
            <xs:element ref="written_name" />
        </xs:sequence>
        <xs:attribute name="card_id" type="xs:integer" use="required" />
        <xs:attribute name="librarian_id" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="borrowerType">
        <xs:sequence>
            <xs:element name="name" type="nameType" />
            <xs:element ref="date_of_birth" />
            <xs:element ref="age" />
            <xs:element name="address" type="addressType" minOccurs="0" />
            <xs:element ref="phone_number" minOccurs="0" maxOccurs="unbounded"
/>
            <xs:element ref="email_address" minOccurs="0"
maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="borrower_id" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="handingOverType">
        <xs:sequence>
            <xs:element ref="handing_date" minOccurs="0" />
            <xs:element ref="lending_or_taking" />
```

```
        </xs:sequence>
        <xs:attribute name="book_id" type="xs:integer" use="required" />
        <xs:attribute name="librarian_id" type="xs:integer" use="required" />
        <xs:attribute name="borrower_id" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="bookType">
        <xs:sequence>
            <xs:element ref="title" />
            <xs:element ref="borrowable" />
        </xs:sequence>
        <xs:attribute name="book_id" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="borrowedByType">
        <xs:sequence>
            <xs:element ref="late_fee" />
            <xs:element ref="deadline" />
        </xs:sequence>
        <xs:attribute name="book_id" type="xs:integer" use="required" />
        <xs:attribute name="borrower_id" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="authorType">
        <xs:sequence>
            <xs:element name="name" type="nameType" />
            <xs:element ref="date_of_birth" />
            <xs:element ref="age" />
        </xs:sequence>
        <xs:attribute name="author_id" type="xs:integer" use="required" />
    </xs:complexType>

    <xs:complexType name="writtenByType">
        <xs:sequence>
            <xs:element ref="publication_date" />
        </xs:sequence>
        <xs:attribute name="book_id" type="xs:integer" use="required" />
        <xs:attribute name="author_id" type="xs:integer" use="required" />
    </xs:complexType>

    <!-- Use of the elements from the root element -->

    <xs:element name="C0LLER_Library">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="librarian" type="librarianType"
minOccurs="3"
                    maxOccurs="unbounded" />
```

```
                <xs:element name="business_card" type="businessCardType"
minOccurs="3"
                    maxOccurs="unbounded" />
                <xs:element name="borrower" type="borrowerType" minOccurs="3"
maxOccurs="unbounded" />
                <xs:element name="handing_over" type="handingOverType"
minOccurs="3"
                    maxOccurs="unbounded" />
                <xs:element name="book" type="bookType" minOccurs="0"
maxOccurs="unbounded" />
                <xs:element name="borrowed_by" type="borrowedByType"
minOccurs="0"
                    maxOccurs="unbounded" />
                <xs:element name="author" type="authorType" minOccurs="0"
maxOccurs="unbounded" />
                <xs:element name="written_by" type="writtenByType"
minOccurs="0"
                    maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>

        <!-- Unique keys -->

        <xs:key name="business_card_key">
            <xs:selector xpath="business_card" />
            <xs:field xpath="@card_id" />
        </xs:key>

        <xs:key name="librarian_key">
            <xs:selector xpath="librarian" />
            <xs:field xpath="@librarian_id" />
        </xs:key>

        <xs:key name="handing_over_key">
            <xs:selector xpath="handing_over" />
            <xs:field xpath="@book_id" />
        </xs:key>

        <xs:key name="borrower_key">
            <xs:selector xpath="borrower" />
            <xs:field xpath="@borrower_id" />
        </xs:key>

        <xs:key name="book_key">
            <xs:selector xpath="book" />
            <xs:field xpath="@book_id" />
        </xs:key>
```

```
    <xs:key name="author_key">
        <xs:selector xpath="author" />
        <xs:field xpath="@author_id" />
    </xs:key>

    <!-- Foreign keys -->

    <xs:keyref name="business_card_librarian_key" refer="librarian_key">
        <xs:selector xpath="business_card" />
        <xs:field xpath="@librarian_id" />
    </xs:keyref>

    <xs:keyref name="handing_over_librarian_key" refer="librarian_key">
        <xs:selector xpath="handing_over" />
        <xs:field xpath="@librarian_id" />
    </xs:keyref>

    <xs:keyref name="handing_over_borrower_key" refer="borrower_key">
        <xs:selector xpath="handing_over" />
        <xs:field xpath="@borrower_id" />
    </xs:keyref>

    <xs:keyref name="borrowed_by_borrower_key" refer="borrower_key">
        <xs:selector xpath="borrowed_by" />
        <xs:field xpath="@borrower_id" />
    </xs:keyref>

    <xs:keyref name="borrowed_by_book_key" refer="book_key">
        <xs:selector xpath="borrowed_by" />
        <xs:field xpath="@book_id" />
    </xs:keyref>

    <xs:keyref name="written_by_book_key" refer="book_key">
        <xs:selector xpath="written_by" />
        <xs:field xpath="@book_id" />
    </xs:keyref>

    <xs:keyref name="written_by_author_key" refer="author_key">
        <xs:selector xpath="written_by" />
        <xs:field xpath="@author_id" />
    </xs:keyref>

    <!-- 1:1 connection -->
    <xs:unique name="business_card_librarian_oneone">
        <xs:selector xpath="business_card" />
        <xs:field xpath="@librarian_id" />
    </xs:unique>
```

```
    </xs:element>

</xs:schema>
```

2. feladat

2a) adatolvasás

Először beolvastam az XML dokumentumot, valamint létrehoztam az output fájlt. Ezt követően kiírattam a tartalmat a konzolra és a fájlba, párhuzamos módon. Első lépésként a prolog-ot és a gyökérelemet írattam ki, ezt követően függvények segítségével lekértem az elemeket és ezeket is kiírattam, majd lezártam a gyökérelemet.
A kiíratásokat két fügvénnyel oldottam meg, az egyik node listákat ír ki (printNodeList), valamint meghívja a másikat (printNode), amely az egyes node-ok kiírásáért felelős. Ezentúl egy segédfüggvényt használtam a behúzások megvalósítására, a struktúrált kiíratás érdekében.

```
package hu.domparse.c0ller;

import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;
import java.util.StringJoiner;

public class DOMReadC0LLER {
    public static void main(String[] args) {
        try {
            // Reading in the file
            File xmlFile = new
File("C:\\Users\\vanda\\Desktop\\XMLTaskC0LLER\\XMLC0LLER.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);
            doc.getDocumentElement().normalize();

            File outputFile = new File("read.xml");
            PrintWriter writer = new PrintWriter(new FileWriter(outputFile,
true));

            // Printing out the root element and the prolog
            Element rootElement = doc.getDocumentElement();
            String rootName = rootElement.getTagName();
            StringJoiner rootAttributes = new StringJoiner(" ");
            NamedNodeMap rootAttributeMap = rootElement.getAttributes();

            for (int i = 0; i < rootAttributeMap.getLength(); i++) {
                Node attribute = rootAttributeMap.item(i);
```

```
                rootAttributes.add(attribute.getNodeName() + "=\"" +
attribute.getNodeValue() + "\"");
            }

            System.out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
            writer.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");

            System.out.println("<" + rootName + " " +
rootAttributes.toString() + ">\n");
            writer.println("<" + rootName + " " + rootAttributes.toString() +
">\n");

            // Processing the elements
            NodeList librarianList = doc.getElementsByTagName("librarian");
            NodeList businessCardList =
doc.getElementsByTagName("business_card");
            NodeList borrowerList = doc.getElementsByTagName("borrower");
            NodeList handingOverList =
doc.getElementsByTagName("handing_over");
            NodeList bookList = doc.getElementsByTagName("book");
            NodeList borrowedByList = doc.getElementsByTagName("borrowed_by");
            NodeList authorList = doc.getElementsByTagName("author");
            NodeList writtenByList = doc.getElementsByTagName("written_by");

            // Printing out the (formatted) XML to the console and to the file
            printNodeList(librarianList, writer);
            printNodeList(businessCardList, writer);
            printNodeList(borrowerList, writer);
            printNodeList(handingOverList, writer);
            printNodeList(bookList, writer);
            printNodeList(borrowedByList, writer);
            printNodeList(authorList, writer);
            printNodeList(writtenByList, writer);

            // Closing the root element
            System.out.println("</" + rootName + ">");
            writer.append("</" + rootName + ">");

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

    // Printing out the content of the NodeList recursively
    private static void printNodeList(NodeList nodeList, PrintWriter writer) {
        for (int i = 0; i < nodeList.getLength(); i++) {
            Node node = nodeList.item(i);
```

```
            printNode(node, 1, writer);
            System.out.println("");
            writer.println("");
        }
        System.out.println("");
        writer.println("");
    }

    // Printing out the content of the Node recursively
    private static void printNode(Node node, int indent, PrintWriter writer) {
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;
            String nodeName = element.getTagName();
            StringJoiner attributes = new StringJoiner(" ");
            NamedNodeMap attributeMap = element.getAttributes();

            // Name and attributes
            for (int i = 0; i < attributeMap.getLength(); i++) {
                Node attribute = attributeMap.item(i);
                attributes.add(attribute.getNodeName() + "=\"" +
attribute.getNodeValue() + "\"");
            }

            System.out.print(getIndentString(indent));
            System.out.print("<" + nodeName + " " + attributes.toString() +
">");

            writer.print(getIndentString(indent));
            writer.print("<" + nodeName + " " + attributes.toString() + ">");

            // Content
            NodeList children = element.getChildNodes();
            if (children.getLength() == 1 && children.item(0).getNodeType() ==
Node.TEXT_NODE) {
                System.out.print(children.item(0).getNodeValue());
                writer.print(children.item(0).getNodeValue());
            } else {
                System.out.println();
                writer.println();
                for (int i = 0; i < children.getLength(); i++) {
                    printNode(children.item(i), indent + 1, writer);
                }
                System.out.print(getIndentString(indent));
                writer.print(getIndentString(indent));
            }

            // Closing the node
            System.out.println("</" + nodeName + ">");
```

```
            writer.println("</" + nodeName + ">");
        }
    }

    // Indenting
    private static String getIndentString(int indent) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < indent; i++) {
            sb.append("  "); // 2 spaces per indent level
        }
        return sb.toString();
    }
}
```

2b) adatmódosítás

Először beolvastam az XML fájlt, majd végrehajtottam a következő módosításokat:

- Egy könyvtáros vezeték- és keresztnevének megváltoztatása
- A névjegykártyára írt név megváltoztatása
- Egy kölcsönző telefonszámának megváltoztatása
- Egy könyv címének a megváltoztatása
- Egy szerző korának megváltoztatása

Ezt követően string-gé alakítottam a módosított XML-t és kiírattam a konzolra, struktúrált formában.

```
package hu.domparse.c0ller;

import org.w3c.dom.*;

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.File;
import java.io.StringWriter;

public class DOMModifyC0LLER {
    public static void main(String[] args) {
        try {
            // Reading in the file
            File xmlFile = new
File("C:\\Users\\vanda\\Desktop\\XMLTaskC0LLER\\XMLC0LLER.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);


            // Implementation of changes

            //Changing the first and last name of a librarian
            NodeList librarianList = doc.getElementsByTagName("librarian");
            Element librarian = (Element) librarianList.item(1);
            librarian.getElementsByTagName("first_name").item(0).setTextConten
t("Nagy");
```

```
            librarian.getElementsByTagName("last_name").item(0).setTextContent
("Gergo");

            //Changing the written name on a business card
            NodeList businessCardList =
doc.getElementsByTagName("business_card");
            Element businessCard = (Element) businessCardList.item(1);
            businessCard.getElementsByTagName("written_name").item(0).setTextC
ontent("Nagy Gergo");

            //Changing the phone number of a borrower
            NodeList borrowerList = doc.getElementsByTagName("borrower");
            Element borrower = (Element) borrowerList.item(0);
            borrower.getElementsByTagName("phone_number").item(1).setTextConte
nt("0620-829-4357");

            //Changing the title of a book
            NodeList bookList = doc.getElementsByTagName("book");
            Element book = (Element) bookList.item(2);
            book.getElementsByTagName("title").item(0).setTextContent("Szamjat
ek");

            //Changing the age of an author
            NodeList authorList = doc.getElementsByTagName("author");
            Element author = (Element) authorList.item(1);
            author.getElementsByTagName("age").item(0).setTextContent("68");


            // Transforming the modified document to a string
            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();

            // Creating a StringWriter to write the XML string
            StringWriter writer = new StringWriter();
            transformer.transform(new DOMSource(doc), new
StreamResult(writer));

            // Getting the modified XML as a string
            String output = writer.toString();
            // Printing out the modified XML to the console
            System.out.println(output);

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

2c) adatlekérdezés

Első lépésként a már megszokott módon itt is beolvastam az XML dokumentumot. Ezután a következő lekérdezéseket valósítottam meg:

- Az összes könyvtáros nevének lekérdezése
- A ’22’-es ID-val rendelkező kölcsönző adatainak lekérdezése
- A 60 éves vagy annál idősebb könyvtárosok nevének lekérdezése
- Az ’Ahol a folyami rákok énekelnek’ című könyv adatainak lekérdezése
- A ’Rácz Maja’ által kikölcsönzött könyv címének lekérdezése

```
package hu.domparse.c0ller;

import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;

public class DOMQueryC0LLER {
    public static void main(String[] args) {
        try {
            // Reading in the file
            File xmlFile = new
File("C:\\Users\\vanda\\Desktop\\XMLTaskC0LLER\\XMLC0LLER.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);

            // Queries
            firstQuery(doc);
            secondQuery(doc);
            thirdQuery(doc);
            fourthQuery(doc);
            fifthQuery(doc);


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

    private static void firstQuery(Document doc)
    {
        // Querying the name of all librarians
        NodeList librarianList = doc.getElementsByTagName("librarian");
        System.out.println("1, Librarians: ");
```

```
        for (int i = 0; i < librarianList.getLength(); i++) {
            Element librarian = (Element) librarianList.item(i);
            String firstName =
librarian.getElementsByTagName("first_name").item(0).getTextContent();
            String lastName =
librarian.getElementsByTagName("last_name").item(0).getTextContent();
            System.out.println(lastName + " " + firstName);
        }
        System.out.println();
    }

    private static void secondQuery(Document doc)
    {
        // Querying the data of the borrower with ID '22'
        String borrowerId = "22";
        NodeList borrowerList = doc.getElementsByTagName("borrower");
        for (int i = 0; i < borrowerList.getLength(); i++) {
            Element borrower = (Element) borrowerList.item(i);
            if (borrower.getAttribute("borrower_id").equals(borrowerId)) {
                String firstName =
borrower.getElementsByTagName("first_name").item(0).getTextContent();
                String lastName =
borrower.getElementsByTagName("last_name").item(0).getTextContent();
                String dateOfBirth =
borrower.getElementsByTagName("date_of_birth").item(0).getTextContent();
                String age =
borrower.getElementsByTagName("age").item(0).getTextContent();
                String city =
borrower.getElementsByTagName("city").item(0).getTextContent();
                String street =
borrower.getElementsByTagName("street").item(0).getTextContent();
                String houseNumber =
borrower.getElementsByTagName("house_number").item(0).getTextContent();

                System.out.println("2, Data of the borrower with ID '" +
borrowerId + "':");
                System.out.println("Name: " + lastName + " " + firstName);
                System.out.println("Date of birth: " + dateOfBirth);
                System.out.println("Age: " + age);
                System.out.println("Address: " + city + ", " + street + " " +
houseNumber);

                NodeList phoneNumberList =
borrower.getElementsByTagName("phone_number");
                System.out.print("Phone number(s): ");
                for (int j = 0; j < phoneNumberList.getLength(); j++) {
                    String phoneNumber =
phoneNumberList.item(j).getTextContent();
```

```
                    System.out.print(phoneNumber);
                    if(j<phoneNumberList.getLength()-1)
                        System.out.print(", ");
                }

                System.out.println();
                Node emailNode =
borrower.getElementsByTagName("email_address").item(0);
                if (emailNode != null) {
                    String emailAddress = emailNode.getTextContent();
                    System.out.println("Email address: " + emailAddress);
                }
                break;
            }
        }
        System.out.println();
    }

    private static void thirdQuery(Document doc)
    {
        // Querying the name of the author(s) who are 60 years old or older
        NodeList authorList = doc.getElementsByTagName("author");
        System.out.println("3, Author(s) who are 60 years old or older:");
        for (int i = 0; i < authorList.getLength(); i++) {
            Element author = (Element) authorList.item(i);
            int age =
Integer.parseInt(author.getElementsByTagName("age").item(0).getTextContent());

            if (age >= 60) {
                String firstName =
author.getElementsByTagName("first_name").item(0).getTextContent();
                String lastName =
author.getElementsByTagName("last_name").item(0).getTextContent();
                System.out.println(firstName + " " + lastName);
            }
        }
        System.out.println();
    }

    private static void fourthQuery(Document doc)
    {
        // Querying the data of the book with the title 'Ahol a folyami rákok
énekelnek'
        String bookTitle = "Ahol a folyami rákok énekelnek";
        boolean isBorrowable = false;

        NodeList bookList = doc.getElementsByTagName("book");
        for (int i = 0; i < bookList.getLength(); i++) {
```

```
            Element book = (Element) bookList.item(i);
            String title =
book.getElementsByTagName("title").item(0).getTextContent();

            if (title.equals(bookTitle)) {
                isBorrowable =
"1".equals(book.getElementsByTagName("borrowable").item(0).getTextContent());
                System.out.println("4, Data of the book: ");
                System.out.println("Title: " + title);
                System.out.println("Borrowable: " + (isBorrowable ? "yes" :
"no"));
                break;
            }
        }
        System.out.println();
    }

    private static void fifthQuery(Document doc)
    {
        // Querying the title of the book borrowed by 'Rácz Maja'
        String borrowerName = "Rácz Maja";

        NodeList borrowerList = doc.getElementsByTagName("borrower");
        for (int i = 0; i < borrowerList.getLength(); i++) {
            Element borrower = (Element) borrowerList.item(i);
            String firstName =
borrower.getElementsByTagName("first_name").item(0).getTextContent();
            String lastName =
borrower.getElementsByTagName("last_name").item(0).getTextContent();
            String borrowerFullName = lastName + " " + firstName;

            if (borrowerFullName.equals(borrowerName)) {
                String borrowerId = borrower.getAttribute("borrower_id");

                NodeList borrowedByList =
doc.getElementsByTagName("borrowed_by");
                System.out.println("5, Title of the book borrowed by " +
borrowerName + ": ");
                for (int j = 0; j < borrowedByList.getLength(); j++) {
                    Element borrowedBy = (Element) borrowedByList.item(j);
                    if
(borrowedBy.getAttribute("borrower_id").equals(borrowerId)) {
                        String bookId = borrowedBy.getAttribute("book_id");

                        NodeList bookList = doc.getElementsByTagName("book");
                        for (int k = 0; k < bookList.getLength(); k++) {
                            Element book = (Element) bookList.item(k);
                            if (book.getAttribute("book_id").equals(bookId)) {
```

```
                                String bookTitle =
book.getElementsByTagName("title").item(0).getTextContent();
                                System.out.println(bookTitle);
                            }
                        }
                    }
                }
                break;
            }
        }
    }
}
```

2d) adatírás

Első lépésként létrehoztam egy új dokumentumot, majd feltöltöttem adatokkal. Ezt egy olyan függvénnyel (addData) oldottam meg, amely létrehozza a prolog-ot és a gyökérelemet, majd meghívja az egyes elemeket létrehozó függvényeket, a különböző adatokkal paraméterként. Ezután a Read-nél alkalmazott módon, struktúrált formában kiírattam a konzolra és egy fájlba az XML dokumentumot.

```
package hu.domparse.c0ller;

import java.io.File;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.util.StringJoiner;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.w3c.dom.NamedNodeMap;

public class DOMWriteC0LLER {
    public static void main(String[] args) {
        try {
             // Creating a new document
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document doc = builder.newDocument();

            // Filling up the document with data
            addData(doc);

            File outputFile = new File("XMLC0LLER1.xml");
            PrintWriter writer = new PrintWriter(new FileWriter(outputFile,
true));

            // Printing out the root element and the prolog
            Element rootElement = doc.getDocumentElement();
            String rootName = rootElement.getTagName();
            StringJoiner rootAttributes = new StringJoiner(" ");
            NamedNodeMap rootAttributeMap = rootElement.getAttributes();

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

```
                Node attribute = rootAttributeMap.item(i);
                rootAttributes.add(attribute.getNodeName() + "=\"" +
attribute.getNodeValue() + "\"");
            }

            System.out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
            writer.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");

            System.out.println("<" + rootName + " " +
rootAttributes.toString() + ">\n");
            writer.println("<" + rootName + " " + rootAttributes.toString() +
">\n");

            // Processing the elements
            NodeList librarianList = doc.getElementsByTagName("librarian");
            NodeList businessCardList =
doc.getElementsByTagName("business_card");
            NodeList borrowerList = doc.getElementsByTagName("borrower");
            NodeList handingOverList =
doc.getElementsByTagName("handing_over");
            NodeList bookList = doc.getElementsByTagName("book");
            NodeList borrowedByList = doc.getElementsByTagName("borrowed_by");
            NodeList authorList = doc.getElementsByTagName("author");
            NodeList writtenByList = doc.getElementsByTagName("written_by");

            // Printing out the (formatted) XML to the console and to the file
            printNodeList(librarianList, writer);
            printNodeList(businessCardList, writer);
            printNodeList(borrowerList, writer);
            printNodeList(handingOverList, writer);
            printNodeList(bookList, writer);
            printNodeList(borrowedByList, writer);
            printNodeList(authorList, writer);
            printNodeList(writtenByList, writer);

            // Closing the root element
            System.out.println("</" + rootName + ">");
            writer.append("</" + rootName + ">");

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

    private static void addData(Document doc) {

        Element rootElement = doc.createElement("C0LLER_Library");
```

```
        rootElement.setAttribute("xmlns:xs",
"http://www.w3.org/2001/XMLSchema-instance");
        rootElement.setAttribute("xs:noNamespaceSchemaLocation",
"XMLSchemaC0LLER.xsd");
        doc.appendChild(rootElement);

        addLibrarian(doc, rootElement, "01", "Klaudia", "Fehér", "2000-05-17",
"23", "0630-080-8126", "0620-871-4015");
        addLibrarian(doc, rootElement, "02", "Albert", "Pintér", "1988-12-17",
"35", "0620-155-0102", null);
        addLibrarian(doc, rootElement, "03", "Erzsébet", "Farkas", "1966-03-
12", "57", "0670-440-8020", null);

        addBusinessCard(doc, rootElement, "11", "01", "Fehér Klaudia");
        addBusinessCard(doc, rootElement, "12", "02", "Pintér Albert");
        addBusinessCard(doc, rootElement, "13", "03", "Farkas Erzsébet");

        addBorrower(doc, rootElement, "21", "Árpád", "Jakab", "1965-01-12",
"58", "Miskolc", "Vitéz utca", "31", "0630-651-4518", "0670-749-3819", null);
        addBorrower(doc, rootElement, "22", "Georgina", "Somogyi", "1972-10-
24", "51", "Arnót", "Kisfaludy utca", "17", "0620-845-4708", null,
"somogyigeorgina@citromail.hu");
        addBorrower(doc, rootElement, "23", "Maja", "Rácz", "2000-11-28",
"23", null, null, null, null, null, "rmaja@freemail.hu");

        addHandingOver(doc, rootElement, "32", "02", "21", null, "1");
        addHandingOver(doc, rootElement, "31", "03", "22", "2023-08-23","1");
        addHandingOver(doc, rootElement, "33", "01", "23", "2022-12-16", "0");

        addBook(doc, rootElement, "31", "Harry Potter és a Bölcsek Köve",
"1");
        addBook(doc, rootElement, "32", "Ahol a folyami rákok énekelnek",
"1");
        addBook(doc, rootElement, "33", "Fordulópont", "1");

        addBorrowedBy(doc, rootElement, "32", "21", "0", "2023-03-17");
        addBorrowedBy(doc, rootElement, "31", "22", "0", "2023-09-01");
        addBorrowedBy(doc, rootElement, "33", "23", "800", "2023-02-16");

        addAuthor(doc, rootElement, "41", "Joanne", "Rowling", "1965-07-31",
"58");
        addAuthor(doc, rootElement, "42", "Delia", "Owens", "1949-04-04",
"74");
        addAuthor(doc, rootElement, "43", "Danielle", "Steel", "1947-08-14",
"76");

        addWrittenBy(doc, rootElement, "31", "41", "2001-12-13");
        addWrittenBy(doc, rootElement, "32", "42", "2018-08-14");
```

```
        addWrittenBy(doc, rootElement, "33", "43", "2019-07-21");
    }

    private static void addLibrarian(Document doc, Element rootElement, String
librarianId, String firstName, String lastName, String dateOfBirth, String
age, String phoneNumber1, String phoneNumber2) {
        Element librarian = doc.createElement("librarian");
        librarian.setAttribute("librarian_id", librarianId);

        Element name = doc.createElement("name");
        Element firstNameElement = createElement(doc, "first_name",
firstName);
        Element lastNameElement = createElement(doc, "last_name", lastName);
        name.appendChild(firstNameElement);
        name.appendChild(lastNameElement);

        Element dateOfBirthElement = createElement(doc, "date_of_birth",
dateOfBirth);
        Element ageElement = createElement(doc, "age", age);

        librarian.appendChild(name);
        librarian.appendChild(dateOfBirthElement);
        librarian.appendChild(ageElement);

        if (phoneNumber1 != null && !phoneNumber1.isEmpty()) {
            Element phoneNumber1Element = createElement(doc, "phone_number",
phoneNumber1);
            librarian.appendChild(phoneNumber1Element);
        }
        if (phoneNumber2 != null && !phoneNumber2.isEmpty()) {
            Element phoneNumber2Element = createElement(doc, "phone_number",
phoneNumber2);
            librarian.appendChild(phoneNumber2Element);
        }

        rootElement.appendChild(librarian);
    }

    private static void addBusinessCard(Document doc, Element rootElement,
String cardId, String librarianId, String writtenName) {
        Element businessCard = doc.createElement("business_card");
        businessCard.setAttribute("card_id", cardId);
        businessCard.setAttribute("librarian_id", librarianId);

        Element writtenNameElement = createElement(doc, "written_name",
writtenName);

        businessCard.appendChild(writtenNameElement);
```

```
        rootElement.appendChild(businessCard);
    }

    private static void addBorrower(Document doc, Element rootElement, String
borrowerId, String firstName, String lastName, String dateOfBirth, String age,
String city, String street, String houseNumber, String phoneNumber1, String
phoneNumber2, String emailAddress) {
        Element borrower = doc.createElement("borrower");
        borrower.setAttribute("borrower_id", borrowerId);

        Element name = doc.createElement("name");
        Element firstNameElement = createElement(doc, "first_name",
firstName);
        Element lastNameElement = createElement(doc, "last_name", lastName);
        name.appendChild(firstNameElement);
        name.appendChild(lastNameElement);

        Element dateOfBirthElement = createElement(doc, "date_of_birth",
dateOfBirth);
        Element ageElement = createElement(doc, "age", age);

        borrower.appendChild(name);
        borrower.appendChild(dateOfBirthElement);
        borrower.appendChild(ageElement);

        if(city != null && !city.isEmpty()) {
            Element address = doc.createElement("address");

            Element cityElement = createElement(doc, "city", city);
            Element streetElement = createElement(doc, "street", street);
            Element houseNumberElement = createElement(doc, "house_number",
houseNumber);

            address.appendChild(cityElement);
            address.appendChild(streetElement);
            address.appendChild(houseNumberElement);

            borrower.appendChild(address);
        }

        if (phoneNumber1 != null && !phoneNumber1.isEmpty()) {
            Element phoneNumber1Element = createElement(doc, "phone_number",
phoneNumber1);
            borrower.appendChild(phoneNumber1Element);
        }
        if (phoneNumber2 != null && !phoneNumber2.isEmpty()) {
```

```
            Element phoneNumber2Element = createElement(doc, "phone_number",
phoneNumber2);
            borrower.appendChild(phoneNumber2Element);
        }

        if (emailAddress != null && !emailAddress.isEmpty()) {
            Element emailAddressElement = createElement(doc, "email_address",
emailAddress);
            borrower.appendChild(emailAddressElement);
        }

        rootElement.appendChild(borrower);
    }

    private static void addHandingOver(Document doc, Element rootElement,
String bookId, String librarianId, String borrowerId, String handingDate,
String lendingOrTaking) {
        Element handingOverSwitchboard = doc.createElement("handing_over");
        handingOverSwitchboard.setAttribute("book_id", bookId);
        handingOverSwitchboard.setAttribute("librarian_id", librarianId);
        handingOverSwitchboard.setAttribute("borrower_id", borrowerId);

        Element lendingOrTakingElement = createElement(doc,
"lending_or_taking", lendingOrTaking);
        Element handingDateElement = createElement(doc, "handing_date",
handingDate);

        handingOverSwitchboard.appendChild(lendingOrTakingElement);
        handingOverSwitchboard.appendChild(handingDateElement);

        rootElement.appendChild(handingOverSwitchboard);
    }

    private static void addBook(Document doc, Element rootElement, String
bookId, String title, String borrowable) {
        Element book = doc.createElement("book");
        book.setAttribute("book_id", bookId);

        Element titleElement = createElement(doc, "title", title);
        Element borrowableElement = createElement(doc, "borrowable",
borrowable);

        book.appendChild(titleElement);
        book.appendChild(borrowableElement);

        rootElement.appendChild(book);
    }
```

```
    private static void addBorrowedBy(Document doc, Element rootElement,
String bookId, String borrowerId, String lateFee, String deadline) {
        Element borrowedBySwitchboard = doc.createElement("borrowed_by");
        borrowedBySwitchboard.setAttribute("book_id", bookId);
        borrowedBySwitchboard.setAttribute("borrower_id", borrowerId);

        Element lateFeeElement = createElement(doc, "late_fee", lateFee);
        Element deadlineElement = createElement(doc, "deadline", deadline);

        borrowedBySwitchboard.appendChild(lateFeeElement);
        borrowedBySwitchboard.appendChild(deadlineElement);

        rootElement.appendChild(borrowedBySwitchboard);
    }

    private static void addAuthor(Document doc, Element rootElement, String
authorId, String firstName, String lastName, String dateOfBirth, String age) {
        Element author = doc.createElement("author");
        author.setAttribute("author_id", authorId);

        Element name = doc.createElement("name");
        Element firstNameElement = createElement(doc, "first_name",
firstName);
        Element lastNameElement = createElement(doc, "last_name", lastName);
        name.appendChild(firstNameElement);
        name.appendChild(lastNameElement);

        Element dateOfBirthElement = createElement(doc, "date_of_birth",
dateOfBirth);
        Element ageElement = createElement(doc, "age", age);

        author.appendChild(name);
        author.appendChild(dateOfBirthElement);
        author.appendChild(ageElement);

        rootElement.appendChild(author);
    }

    private static void addWrittenBy(Document doc, Element rootElement, String
bookId, String authorId, String publicationDate) {
        Element writtenBySwitchboard = doc.createElement("written_by");
        writtenBySwitchboard.setAttribute("book_id", bookId);
        writtenBySwitchboard.setAttribute("author_id", authorId);

        Element publicationDateElement = createElement(doc,
"publication_date", publicationDate);

        writtenBySwitchboard.appendChild(publicationDateElement);
```

```
        rootElement.appendChild(writtenBySwitchboard);
    }

    private static Element createElement(Document doc, String name, String
value) {
        Element element = doc.createElement(name);
        element.appendChild(doc.createTextNode(value));
        return element;
    }

    // Printing out the content of the NodeList recursively
    private static void printNodeList(NodeList nodeList, PrintWriter writer) {
        for (int i = 0; i < nodeList.getLength(); i++) {
          Node node = nodeList.item(i);
          printNode(node, 1, writer);
          System.out.println("");
          writer.println("");
        }
        System.out.println("");
        writer.println("");
    }

    // Printing out the content of the Node recursively
    private static void printNode(Node node, int indent, PrintWriter writer) {
        if (node.getNodeType() == Node.ELEMENT_NODE) {
          Element element = (Element) node;
          String nodeName = element.getTagName();
          StringJoiner attributes = new StringJoiner(" ");
          NamedNodeMap attributeMap = element.getAttributes();

          // Name and attributes
          for (int i = 0; i < attributeMap.getLength(); i++) {
            Node attribute = attributeMap.item(i);
            attributes.add(attribute.getNodeName() + "=\"" +
attribute.getNodeValue() + "\"");
          }

          System.out.print(getIndentString(indent));
          System.out.print("<" + nodeName + " " + attributes.toString()+ ">");

          writer.print(getIndentString(indent));
          writer.print("<" + nodeName + " " + attributes.toString() + ">");

          // Content
          NodeList children = element.getChildNodes();
          if (children.getLength() == 1 && children.item(0).getNodeType() ==
Node.TEXT_NODE) {
```

```
            System.out.print(children.item(0).getNodeValue());
            writer.print(children.item(0).getNodeValue());
          } else {
            System.out.println();
            writer.println();
            for (int i = 0; i < children.getLength(); i++) {
              printNode(children.item(i), indent + 1, writer);
            }
            System.out.print(getIndentString(indent));
            writer.print(getIndentString(indent));
          }

          // Closing the node
          System.out.println("</" + nodeName + ">");
          writer.println("</" + nodeName + ">");
        }
    }

    // Indenting
    private static String getIndentString(int indent) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < indent; i++) {
          sb.append("  "); // 2 spaces per indent level
        }
        return sb.toString();
    }
}
```