JEGYZŐKÖNYV

Adatkezelés XML környezetben

Féléves feladat

Kórház

KészítetteTihor Fruzsina

Neptunkód: THDWDR

Dátum:2023.12.12.

Tartalom

Bevezetés 3
a. A feladat leírása: 3
1. feladat 3
b. Az adatbázis ER modell tervezése 3
c. Az adatbázis konvertálása XDM modellre 4
d. Az XDM modell alapján XML dokumentum készítése: 4
e. Az XML dokumentum alapján XMLSchema készítése 7
2.feladat 12
f. DOM adatolvasás 12
g. DOM adatmódosítás 19
h. DOM adatlekérdezés 21
i. DOM adatírás 24

Bevezetés

a. A feladat leírása:

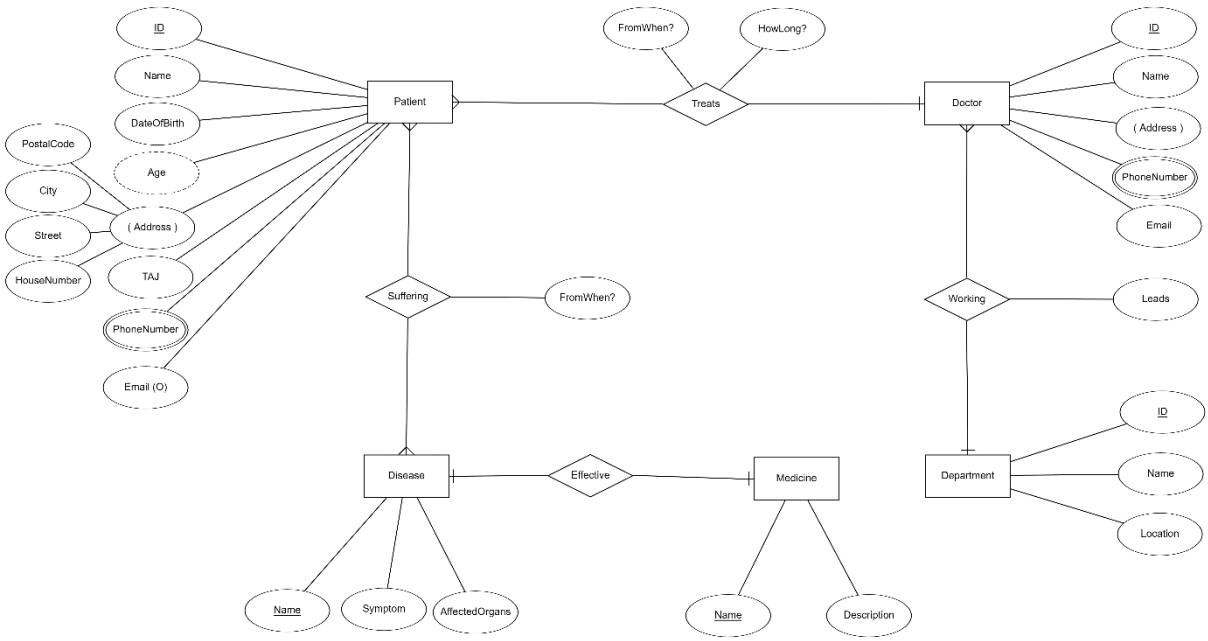
A megvalósítani kívánt adatbázisom egy kórház működésének rendszerét mutatja be. Minden egyes kórházban számos orvos dolgozik. Szükséges eltárolni az orvosok egyedi azonosítóit, amik segítenek megkülönböztetni őket, nevüket, lakcímüket, valamint az elérési lehetőségeket (email, telefonszám). Az orvosok az orvostudomány valamely ágának szakemberei, így valamilyen osztályhoz tartoznak. Az osztályoknak megvannak az azonosítóik, saját nevük, valamint, hogy kórházon belül hol helyezkednek el.

Számos beteget fogadnak a kórházak, minden felvételkor rögzítik a beteg személyes adatait. Ilyenkor kerül be a páciens a rendszerbe, kap egy egyedi azonosítót, rögzítik a nevét, születési dátumát, lakcímét, korát (amit a születési dátum alapján meg is lehet határozni) TAJ számát, valamint a elérési lehetőségeit. Az adatok rögzítése után a betegeket megvizsgálja az orvos és eldönti, hogy milyen betegségben szenved, valamint, hogy kezelésre, gyógyszerre szorul. Feljegyzik, hogy mióta szenvedhet ebben a betegségben a személy, a betegség adatait, valamint, hogy erre milyen gyógyszert és milyen dózisban kell szedni.

Az ER modell tartalmazza a fent említett egyedeket és tulajdonságokat. Az ER modellhez tartozik egy XDM modell amely megadja a működéshez szükséges XML keretet. Az XML-be fel kell vinni az egyedek adatait. Az XML-hez továbbá tartozik egy XMLSchema amely pontosan megadja hogy milyen típusú adatok vihetők fel az XML-be és hogy ezekből mennyit vihetünk fel. Ezek mellé készült egy DOM program, amely a módosításokkal és lekérdezésekkel foglalkozik éppen azzal, amelyikre az adott helyzetben szükségünk van.

1. feladat

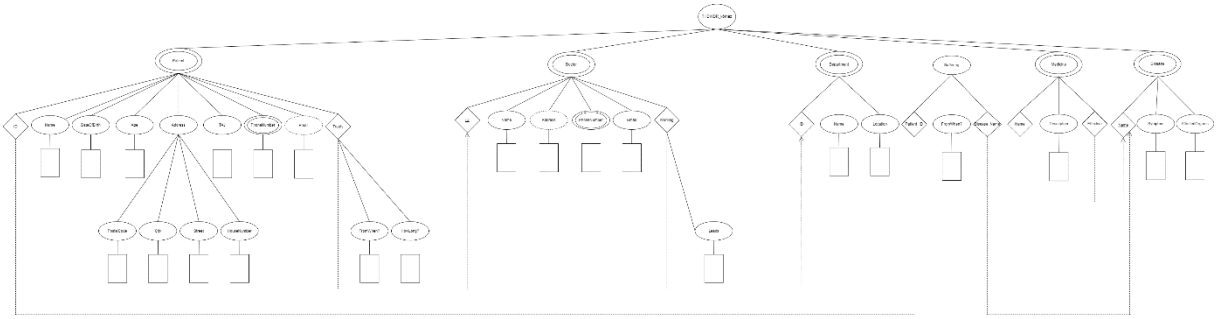
b. Az adatbázis ER modell tervezése



c. *Az adatbázis konvertálása XDM modellre*

XDM modellnél háromféle jelölést alkalmazhatunk. Ezek az ellipszis, a rombusz, illetve a téglalap. Az ellipszis jelöli az elemeket minden egyedből elem lesz, ezen felül a tulajdonságokból is. A rombusz jelöli az attribútumokat, amelyek a kulcs tulajdonságokból keletkeznek. A téglalap jelöli a szöveget, amely majd az XML dokumentumban fog megjelenni. Azoknak az elemeknek, amelyek többször is előfordulhatnak, a jelölése dupla ellipszissel történik. Az idegenkulcsok és a kulcsok közötti kapcsolatot szaggatott vonalas nyíllal jelöljük.

A feladat XDM modellje:



d. *Az XDM modell alapján XML dokumentum készítése:*

Az XDM modell alapján az XML dokumentumot úgy készítettem el, hogy először is a root elementtel kezdtem, ami az THDWDR_korhaz volt. A gyermek elemeiből 3-3 példányt hoztam létre, ezeknek az elemeknek az attribútumai közé tartoznak a kulcsok, illetve idegenkulcsok is, mindezek után ezeknek az elemeknek létrehoztam a többi gyermek elementet is.

XML dokumentum forráskódja:

```
<?xml version="1.0" encoding="UTF-8"?>
<THDWDR_kórház>
    <!-- Patient entities -->
    <Patient>
        <ID>1</ID>
        <Name>Kiss Pista</Name>
        <DateOfBirth>1985-05-10</DateOfBirth>
        <Age>38</Age>
        <Address>
            <PostalCode>1111</PostalCode>
            <City>Budapest</City>
            <Street>Kossuth utca</Street>
            <HouseNumber>5</HouseNumber>
        </Address>
        <TAJ>123456789</TAJ>
        <PhoneNumber>123-456-7890</PhoneNumber>
        <PhoneNumber>987-654-3210</PhoneNumber>
        <Email>kiss.pista@example.com</Email>
    </Patient>

    <Patient>
        <ID>2</ID>
        <Name>Magyari Ádám</Name>
        <DateOfBirth>1992-09-22</DateOfBirth>
        <Age>30</Age>
        <Address>
            <PostalCode>2222</PostalCode>
```

```
        <City>Debrecen</City>
        <Street>Petőfi utca</Street>
        <HouseNumber>12</HouseNumber>
    </Address>
    <TAJ>987654321</TAJ>
    <PhoneNumber>111-222-3333</PhoneNumber>
    <PhoneNumber>444-555-6666</PhoneNumber>
    <Email>magyari.adam@example.com</Email>
</Patient>

<Patient>
    <ID>3</ID>
    <Name>Pácza Laura</Name>
    <DateOfBirth>1980-03-15</DateOfBirth>
    <Age>42</Age>
    <Address>
        <PostalCode>3333</PostalCode>
        <City>Szeged</City>
        <Street>Arany János utca</Street>
        <HouseNumber>8</HouseNumber>
    </Address>
    <TAJ>111222333</TAJ>
    <PhoneNumber>555-666-7777</PhoneNumber>
    <Email>pacza.laura@example.com</Email>
</Patient>

<!-- Doctor entities -->
<Doctor>
    <ID>101</ID>
    <Name>Dr. Kovács Tamás</Name>
    <Address>Szakrendelő</Address>
    <PhoneNumber>111-222-3333</PhoneNumber>
    <PhoneNumber>444-555-6666</PhoneNumber>
    <Email>dr.kovacs.tamas@example.com</Email>
</Doctor>

<Doctor>
    <ID>102</ID>
    <Name>Dr. Nagy Éva</Name>
    <Address>Kórház</Address>
    <PhoneNumber>777-888-9999</PhoneNumber>
    <Email>dr.nagy.eva@example.com</Email>
</Doctor>

<Doctor>
    <ID>103</ID>
    <Name>Dr. Balogh Gábor</Name>
    <Address>Rendelőintézet</Address>
    <PhoneNumber>333-444-5555</PhoneNumber>
    <Email>dr.balogh.gabor@example.com</Email>
</Doctor>

<!-- Department entities -->
<Department>
    <ID>201</ID>
    <Name>Belgyógyászat</Name>
    <Location>2. emelet</Location>
</Department>

<Department>
    <ID>202</ID>
    <Name>Radiológia</Name>
    <Location>4. emelet</Location>
```

```
</Department>

<Department>
    <ID>203</ID>
    <Name>Sebészet</Name>
    <Location>1. emelet</Location>
</Department>

<!-- Medicine entities -->
<Medicine>
    <Name>Paracetamol</Name>
    <Description>Fájdalomcsillapító</Description>
</Medicine>

<Medicine>
    <Name>Antibiotikum</Name>
    <Description>Gyulladáscsökkentő</Description>
</Medicine>

<Medicine>
    <Name>Vitamin C</Name>
    <Description>Immunerősítő</Description>
</Medicine>

<!-- Disease entities -->
<Disease>
    <Name>Influenza</Name>
    <Symptom>Láz, H köhögés</Symptom>
    <AffectedOrgans>Légzőrendszer</AffectedOrgans>
</Disease>

<Disease>
    <Name>Magas vérnyomás</Name>
    <Symptom>Fejfájás, Szédülés</Symptom>
    <AffectedOrgans>Szív-érrendszer</AffectedOrgans>
</Disease>

<Disease>
    <Name>Cukorbetegség</Name>
    <Symptom>Fáradtság, Szomjúság</Symptom>
    <AffectedOrgans>Endokrin rendszer</AffectedOrgans>
</Disease>

<!-- Relationships -->
<Suffering>
    <DiseaseID>1</DiseaseID>
    <PatientID>1</PatientID>
    <FromWhen>2023-02-15</FromWhen>
</Suffering>

<Suffering>
    <DiseaseID>2</DiseaseID>
    <PatientID>2</PatientID>
    <FromWhen>2023-03-20</FromWhen>
</Suffering>

<Suffering>
    <DiseaseID>3</DiseaseID>
    <PatientID>3</PatientID>
    <FromWhen>2023-01-10</FromWhen>
</Suffering>

<Effective>
```

```
        <DiseaseID>1</DiseaseID>
        <MedicineName>Paracetamol</MedicineName>
    </Effective>

    <Effective>
        <DiseaseID>2</DiseaseID>
        <MedicineName>Antibiotikum</MedicineName>
    </Effective>

    <Effective>
        <DiseaseID>3</DiseaseID>
        <MedicineName>Vitamin C</MedicineName>
    </Effective>

    <Treats>
        <DoctorID>101</DoctorID>
        <PatientID>1</PatientID>
        <FromWhen>2023-02-15</FromWhen>
        <HowLong>10 nap</HowLong>
    </Treats>

    <Treats>
        <DoctorID>102</DoctorID>
        <PatientID>2</PatientID>
        <FromWhen>2023-03-20</FromWhen>
        <HowLong>14 nap</HowLong>
    </Treats>

    <Treats>
        <DoctorID>103</DoctorID>
        <PatientID>3</PatientID>
        <FromWhen>2023-01-10</FromWhen>
        <HowLong>7 nap</HowLong>
    </Treats>

    <Working>
        <DoctorID>101</DoctorID>
        <DepartmentID>201</DepartmentID>
        <Leads>true</Leads>
    </Working>

    <Working>
        <DoctorID>102</DoctorID>
        <DepartmentID>202</DepartmentID>
        <Leads>false</Leads>
    </Working>

    <Working>
        <DoctorID>103</DoctorID>
        <DepartmentID>203</DepartmentID>
        <Leads>true</Leads>
    </Working>
</THDWDR_kórház>
```

e. Az XML dokumentum alapján XMLSchema készítése

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

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

```
<!-- Egyszerű típusok definiálása -->
<xs:simpleType name="stringArray">
   <xs:list itemType="xs:string"/>
</xs:simpleType>

<xs:simpleType name="dateType">
   <xs:restriction base="xs:date"/>
</xs:simpleType>

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

<!-- Komplex típusok definiálása -->
<xs:complexType name="AddressType">
   <xs:sequence>
      <xs:element name="PostalCode" type="xs:string"/>
      <xs:element name="City" type="xs:string"/>
      <xs:element name="Street" type="xs:string"/>
      <xs:element name="HouseNumber" type="xs:integer"/>
   </xs:sequence>
</xs:complexType>

<xs:complexType name="PatientType">
   <xs:sequence>
      <xs:element name="ID" type="xs:integer"/>
      <xs:element name="Name" type="xs:string"/>
```

```
        <xs:element name="DateOfBirth" type="dateType"/>
        <xs:element name="Age" type="xs:integer"/>
        <xs:element name="Address" type="AddressType"/>
        <xs:element name="TAJ" type="TAJType"/>
        <xs:element name="PhoneNumber" type="stringArray"/>
        <xs:element name="Email" type="xs:string" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="DoctorType">
    <xs:sequence>
        <xs:element name="ID" type="xs:integer"/>
        <xs:element name="Name" type="xs:string"/>
        <xs:element name="Address" type="xs:string"/>
        <xs:element name="PhoneNumber" type="stringArray"/>
        <xs:element name="Email" type="xs:string"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="DepartmentType">
    <xs:sequence>
        <xs:element name="ID" type="xs:integer"/>
        <xs:element name="Name" type="xs:string"/>
        <xs:element name="Location" type="xs:string"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="MedicineType">
    <xs:sequence>
        <xs:element name="Name" type="xs:string"/>
```

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

<xs:complexType name="DiseaseType">
    <xs:sequence>
        <xs:element name="Name" type="xs:string"/>
        <xs:element name="Symptom" type="xs:string"/>
        <xs:element name="AffectedOrgans" type="xs:string"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="SufferingType">
    <xs:sequence>
        <xs:element name="DiseaseID" type="xs:integer"/>
        <xs:element name="PatientID" type="xs:integer"/>
        <xs:element name="FromWhen" type="dateType"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="EffectiveType">
    <xs:sequence>
        <xs:element name="DiseaseID" type="xs:integer"/>
        <xs:element name="MedicineName" type="xs:string"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="TreatsType">
    <xs:sequence>
        <xs:element name="DoctorID" type="xs:integer"/>
```

```
        <xs:element name="PatientID" type="xs:integer"/>
        <xs:element name="FromWhen" type="dateType"/>
        <xs:element name="HowLong" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="WorkingType">
      <xs:sequence>
         <xs:element name="DoctorID" type="xs:integer"/>
         <xs:element name="DepartmentID" type="xs:integer"/>
         <xs:element name="Leads" type="boolType"/>
      </xs:sequence>
   </xs:complexType>

   <!-- Gyökérelemtől az elemek felhasználása -->
   <xs:element name="THDWDR_kórház">
      <xs:complexType>
         <xs:sequence>
            <!-- Entities -->
            <xs:element name="Patient" type="PatientType" minOccurs="3"
maxOccurs="unbounded"/>
            <xs:element name="Doctor" type="DoctorType" minOccurs="3"
maxOccurs="unbounded"/>
            <xs:element name="Department" type="DepartmentType" minOccurs="3"
maxOccurs="unbounded"/>
            <xs:element name="Medicine" type="MedicineType" minOccurs="3"
maxOccurs="unbounded"/>
            <xs:element name="Disease" type="DiseaseType" minOccurs="3"
maxOccurs="unbounded"/>

            <!-- Relationships -->
            <xs:element name="Suffering" type="SufferingType" minOccurs="3"
maxOccurs="unbounded"/>
```

```
        <xs:element name="Effective" type="EffectiveType" minOccurs="3"
maxOccurs="unbounded"/>

        <xs:element name="Treats" type="TreatsType" minOccurs="3"
maxOccurs="unbounded"/>

        <xs:element name="Working" type="WorkingType" minOccurs="3"
maxOccurs="unbounded"/>

      </xs:sequence>

    </xs:complexType>

  </xs:element>

</xs:schema>
```

2.feladat

f. DOM adatolvasás

readPatient(Document doc): Ez a függvény beolvassa a betegek adatait a dokumentumból, létrehozva és visszaadva egy listát a Patient objektumokkal, minden beteghez egy-egy objektumot.

readDoctor(Document doc): A függvény beolvassa az orvosok adatait a dokumentumból, létrehozva és visszaadva egy listát a Doctor objektumokkal, minden orvoshoz egy-egy objektumot.

readDepartment(Document doc): Beolvassa az osztályok adatait a dokumentumból, létrehozva és visszaadva egy listát a Department objektumokkal, minden osztályhoz egy-egy objektumot.

readMedicine(Document doc): Beolvassa a gyógyszerek adatait a dokumentumból, létrehozva és visszaadva egy listát a Medicine objektumokkal, minden gyógyszerhez egy-egy objektumot.

readDisease(Document doc): A függvény beolvassa a betegségek adatait a dokumentumból, létrehozva és visszaadva egy listát a Disease objektumokkal, minden betegséghez egy-egy objektumot.

readSuffering(Document doc): Beolvassa a szenvedélyek adatait a dokumentumból, létrehozva és visszaadva egy listát a Suffering objektumokkal, minden szenvedéshez egy-egy objektumot.

readEffective(Document doc): A függvény beolvassa a hatékony kezeléseket a dokumentumból, létrehozva és visszaadva egy listát az Effective objektumokkal, minden hatásos kezeléshez egy-egy objektumot.

readTreats(Document doc): Beolvassa a kezeléseket a dokumentumból, létrehozva és visszaadva egy listát a Treats objektumokkal, minden kezeléshez egy-egy objektumot.

readWorking(Document doc): A függvény beolvassa az orvosok munkáját a dokumentumból, létrehozva és visszaadva egy listát a Working objektumokkal, minden munkához egy-egy objektumot.

Minden függvény elemzi a megadott XML dokumentumot és létrehoz egy listát a megfelelő entitásokból, amelyekkel később könnyen dolgozhatunk a Java alkalmazásban.

A forráskód:

```
package hu.domparse.thdwdr;
import org.w3c.dom.*;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.File;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

public class DOMReadTHDWDR {
       public static void main(String[] args) {
           try {
               File inputFile = new File("XMLTHDWDR.xml");
               DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
               DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
               Document doc = dBuilder.parse(inputFile);
               doc.getDocumentElement().normalize();

               // Adatolvasás kezdete
               List<Patient> patients = readPatient(doc);
               List<Doctor> doctors = readDoctor(doc);
               List<Department> departments = readDepartment(doc);
               List<Medicine> medicines = readMedicine(doc);
               List<Disease> diseases = readDisease(doc);
               List<Suffering> sufferings = readSuffering(doc);
               List<Effective> effectives = readEffective(doc);
               List<Treats> treats = readTreats(doc);
               List<Working> workings = readWorking(doc);

               // Most itt lehet dolgozni az adatokkal
               System.out.println("Patients:");
               for (Patient patient : patients) {
                   System.out.println(patient);
               }

               System.out.println("\nDoctors:");
               for (Doctor doctor : doctors) {
                   System.out.println(doctor);
               }

               System.out.println("\nDepartments:");
               for (Department department : departments) {
                   System.out.println(department);
               }

               System.out.println("\nMedicines:");
               for (Medicine medicine : medicines) {
                   System.out.println(medicine);
               }

               System.out.println("\nDiseases:");
               for (Disease disease : diseases) {
                   System.out.println(disease);
```

```
            }

            System.out.println("\nSufferings:");
            for (Suffering suffering : sufferings) {
                System.out.println(suffering);
            }

            System.out.println("\nEffectives:");
            for (Effective effective : effectives) {
                System.out.println(effectives);
            }

            System.out.println("\nTreats:");
            for (Treats treat : treats) {
                System.out.println(treat);
            }

            System.out.println("\nWorking:");
            for (Working working : workings) {
                System.out.println(working);
            }

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


    private static List<Patient> readPatient(Document doc) {
        List<Patient> patients = new ArrayList<>();
        NodeList patientList = doc.getElementsByTagName("Patient");
        for (int temp = 0; temp < patientList.getLength(); temp++) {
            Node patientNode = patientList.item(temp);
            if (patientNode.getNodeType() == Node.ELEMENT_NODE) {
                Element patientElement = (Element) patientNode;
                Patient patient = new Patient();

                // Betöltés a Patient entitásból

patient.setId(patientElement.getElementsByTagName("ID").item(0).getTextContent());

patient.setName(patientElement.getElementsByTagName("Name").item(0).getTextContent
());

patient.setDateOfBirth(patientElement.getElementsByTagName("DateOfBirth").item(0).
getTextContent());

patient.setAge(patientElement.getElementsByTagName("Age").item(0).getTextContent()
);

                // Betöltés az Address entitásból
                Element addressElement = (Element)
patientElement.getElementsByTagName("Address").item(0);

patient.setPostalCode(addressElement.getElementsByTagName("PostalCode").item(0).ge
tTextContent());

patient.setCity(addressElement.getElementsByTagName("City").item(0).getTextContent
());
```

```
patient.setStreet(addressElement.getElementsByTagName("Street").item(0).getTextCon
tent());

patient.setHouseNumber(addressElement.getElementsByTagName("HouseNumber").item(0).
getTextContent());


patient.setTaj(patientElement.getElementsByTagName("TAJ").item(0).getTextContent()
);

                // Telefonok betöltése
                NodeList phoneNumberList =
patientElement.getElementsByTagName("PhoneNumber");
                List<String> phoneNumbers = new ArrayList<>();
                for (int i = 0; i < phoneNumberList.getLength(); i++) {
                    String phoneNumber = phoneNumberList.item(i).getTextContent();
                    phoneNumbers.add(phoneNumber);
                }
                patient.setPhoneNumbers(phoneNumbers);


patient.setEmail(patientElement.getElementsByTagName("Email").item(0).getTextConte
nt());

                patients.add(patient);
            }
        }
        return patients;
    }
    private static List<Doctor> readDoctor(Document doc) {
        List<Doctor> doctors = new ArrayList<>();
        NodeList doctorList = doc.getElementsByTagName("Doctor");
        for (int temp = 0; temp < doctorList.getLength(); temp++) {
            Node doctorNode = doctorList.item(temp);
            if (doctorNode.getNodeType() == Node.ELEMENT_NODE) {
                Element doctorElement = (Element) doctorNode;
                Doctor doctor = new Doctor();

                // Betöltés a Doctor entitásból

doctor.setId(doctorElement.getElementsByTagName("ID").item(0).getTextContent());

doctor.setName(doctorElement.getElementsByTagName("Name").item(0).getTextContent()
);

doctor.setAddress(doctorElement.getElementsByTagName("Address").item(0).getTextCon
tent());

                // Telefonok betöltése
                NodeList phoneNumberList =
doctorElement.getElementsByTagName("PhoneNumber");
                List<String> phoneNumbers = new ArrayList<>();
                for (int i = 0; i < phoneNumberList.getLength(); i++) {
                    String phoneNumber = phoneNumberList.item(i).getTextContent();
                    phoneNumbers.add(phoneNumber);
                }
                doctor.setPhoneNumbers(phoneNumbers);
```

```
doctor.setEmail(doctorElement.getElementsByTagName("Email").item(0).getTextContent
());

                doctors.add(doctor);
            }
        }
        return doctors;
    }
    private static List<Department> readDepartment(Document doc) {
        List<Department> departments = new ArrayList<>();
        NodeList departmentList = doc.getElementsByTagName("Department");
        for (int temp = 0; temp < departmentList.getLength(); temp++) {
            Node departmentNode = departmentList.item(temp);
            if (departmentNode.getNodeType() == Node.ELEMENT_NODE) {
                Element departmentElement = (Element) departmentNode;
                Department department = new Department();

                // Betöltés a Department entitásból

department.setId(departmentElement.getElementsByTagName("ID").item(0).getTextConte
nt());

department.setName(departmentElement.getElementsByTagName("Name").item(0).getTextC
ontent());

department.setLocation(departmentElement.getElementsByTagName("Location").item(0).
getTextContent());

                departments.add(department);
            }
        }
        return departments;
    }
    private static List<Medicine> readMedicine(Document doc) {
        List<Medicine> medicines = new ArrayList<>();
        NodeList medicineList = doc.getElementsByTagName("Medicine");
        for (int temp = 0; temp < medicineList.getLength(); temp++) {
            Node medicineNode = medicineList.item(temp);
            if (medicineNode.getNodeType() == Node.ELEMENT_NODE) {
                Element medicineElement = (Element) medicineNode;
                Medicine medicine = new Medicine();

                // Betöltés a Medicine entitásból

medicine.setName(medicineElement.getElementsByTagName("Name").item(0).getTextConte
nt());

medicine.setDescription(medicineElement.getElementsByTagName("Description").item(0
).getTextContent());

                medicines.add(medicine);
            }
        }
        return medicines;
    }
    private static List<Disease> readDisease(Document doc) {
        List<Disease> diseases = new ArrayList<>();
        NodeList diseaseList = doc.getElementsByTagName("Disease");
```

```
        for (int temp = 0; temp < diseaseList.getLength(); temp++) {
            Node diseaseNode = diseaseList.item(temp);
            if (diseaseNode.getNodeType() == Node.ELEMENT_NODE) {
                Element diseaseElement = (Element) diseaseNode;
                Disease disease = new Disease();

                // Betöltés a Disease entitásból

disease.setName(diseaseElement.getElementsByTagName("Name").item(0).getTextContent
());

disease.setSymptom(diseaseElement.getElementsByTagName("Symptom").item(0).getTextC
ontent());

disease.setAffectedOrgans(diseaseElement.getElementsByTagName("AffectedOrgans").it
em(0).getTextContent());

                diseases.add(disease);
            }
        }
        return diseases;
    }
    private static List<Suffering> readSuffering(Document doc) {
        List<Suffering> sufferings = new ArrayList<>();
        NodeList sufferingList = doc.getElementsByTagName("Suffering");
        for (int temp = 0; temp < sufferingList.getLength(); temp++) {
            Node sufferingNode = sufferingList.item(temp);
            if (sufferingNode.getNodeType() == Node.ELEMENT_NODE) {
                Element sufferingElement = (Element) sufferingNode;
                Suffering suffering = new Suffering();

                // Convert String to int for DiseaseID and PatientID

suffering.setDiseaseID(Integer.parseInt(sufferingElement.getElementsByTagName("Dis
easeID").item(0).getTextContent()));

suffering.setPatientID(Integer.parseInt(sufferingElement.getElementsByTagName("Pat
ientID").item(0).getTextContent()));

                // Convert String to LocalDate for FromWhen

suffering.setFromWhen(LocalDate.parse(sufferingElement.getElementsByTagName("FromW
hen").item(0).getTextContent()));

                sufferings.add(suffering);
            }
        }
        return sufferings;
    }
    private static List<Effective> readEffective(Document doc) {
        List<Effective> effectives = new ArrayList<>();
        NodeList effectiveList = doc.getElementsByTagName("Effective");

        // Effective elemek feldolgozása
        for (int temp = 0; temp < effectiveList.getLength(); temp++) {
            Node effectiveNode = effectiveList.item(temp);
            if (effectiveNode.getNodeType() == Node.ELEMENT_NODE) {
                Element effectiveElement = (Element) effectiveNode;
                Effective effective = new Effective();
```

```
                // DiseaseID konvertálása int típussá
effective.setDiseaseID(Integer.parseInt(effectiveElement.getElementsByTagName("Dis
easeID").item(0).getTextContent()));

                // MedicineName beolvasása
effective.setMedicineName(effectiveElement.getElementsByTagName("MedicineName").it
em(0).getTextContent());

                effectives.add(effective);
            }
        }
        return effectives;
    }

    private static List<Treats> readTreats(Document doc) {
        List<Treats> treats = new ArrayList<>();
        NodeList treatsList = doc.getElementsByTagName("Treats");

        // Treats elemek feldolgozása
        for (int temp = 0; temp < treatsList.getLength(); temp++) {
            Node treatsNode = treatsList.item(temp);
            if (treatsNode.getNodeType() == Node.ELEMENT_NODE) {
                Element treatsElement = (Element) treatsNode;
                Treats treat = new Treats();

                // DoctorID és PatientID konvertálása int típussá
treat.setDoctorID(Integer.parseInt(treatsElement.getElementsByTagName("DoctorID").
item(0).getTextContent()));

treat.setPatientID(Integer.parseInt(treatsElement.getElementsByTagName("PatientID"
).item(0).getTextContent()));

                // FromWhen konvertálása LocalDate típussá
treat.setFromWhen(LocalDate.parse(treatsElement.getElementsByTagName("FromWhen").i
tem(0).getTextContent()));

                // HowLong beolvasása
treat.setHowLong(treatsElement.getElementsByTagName("HowLong").item(0).getTextCont
ent());

                treats.add(treat);
            }
        }
        return treats;
    }

    private static List<Working> readWorking(Document doc) {
        List<Working> workings = new ArrayList<>();
        NodeList workingList = doc.getElementsByTagName("Working");

        // Working elemek feldolgozása
        for (int temp = 0; temp < workingList.getLength(); temp++) {
            Node workingNode = workingList.item(temp);
```

```
            if (workingNode.getNodeType() == Node.ELEMENT_NODE) {
                Element workingElement = (Element) workingNode;
                Working working = new Working();

                // DoctorID és DepartmentID konvertálása int típussá

working.setDoctorID(Integer.parseInt(workingElement.getElementsByTagName("DoctorID
").item(0).getTextContent()));

working.setDepartmentID(Integer.parseInt(workingElement.getElementsByTagName("Depa
rtmentID").item(0).getTextContent()));

                // Leads beolvasása

working.setLeads(Boolean.parseBoolean(workingElement.getElementsByTagName("Leads")
.item(0).getTextContent()));

                workings.add(working);
            }
        }
        return workings;
    }

}
```

g. DOM adatmódosítás

A DOMModifyTHDWDR osztály egy XML dokumentumot olvas be, módosít néhány adatot, majd kiírja az eredményt. Az modifyDoctors függvény eltávolítja az orvosok nevéből a "Dr. " előtagot. A modifyPatientBirthYear függvény megváltoztatja a 2-es azonosítójú páciens születési évét, hozzáfűzve egy évvel. Az addNewDoctor függvény létrehoz egy új orvos elemet és azt hozzáadja az XML-hez. Végül a printDocument függvény kiírja az eredményt XML formátumban. Az osztály egy main függvényt is tartalmaz, ami a fenti függvényeket hívja meg egy példa XML dokumentumon.

A forráskód:

```
package hu.domparse.thdwdr;

import org.w3c.dom.*;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import java.io.File;
import java.io.StringWriter;
import java.time.LocalDate;
import java.util.List;

public class DOMModifyTHDWDR {

    public static void main(String[] args) {
        try {
            Document doc = readXmlDocument("XMLTHDWDR.xml");
```

```
            // Módosítások
            modifyDoctors(doc);
            modifyPatientBirthYear(doc);
            addNewDoctor(doc);

            // Kiírás XML formában
            printDocument(doc);

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

    private static Document readXmlDocument(String fileName) throws Exception {
        File inputFile = new File(fileName);
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        return dBuilder.parse(inputFile);
    }

    private static void modifyDoctors(Document doc) {
        NodeList doctorList = doc.getElementsByTagName("Doctor");

        for (int i = 0; i < doctorList.getLength(); i++) {
            Node doctorNode = doctorList.item(i);
            if (doctorNode.getNodeType() == Node.ELEMENT_NODE) {
                Element doctorElement = (Element) doctorNode;
                String doctorId =
doctorElement.getElementsByTagName("ID").item(0).getTextContent();

                // Módosítjuk az orvosok nevét, példa: Minden "Dr. " előtagot
eltávolítunk
                String doctorName =
doctorElement.getElementsByTagName("Name").item(0).getTextContent();

doctorElement.getElementsByTagName("Name").item(0).setTextContent(doctorName.repla
ce("Dr. ", ""));
            }
        }
    }

    private static void modifyPatientBirthYear(Document doc) {
        NodeList patientList = doc.getElementsByTagName("Patient");

        for (int i = 0; i < patientList.getLength(); i++) {
            Node patientNode = patientList.item(i);
            if (patientNode.getNodeType() == Node.ELEMENT_NODE) {
                Element patientElement = (Element) patientNode;
                String patientId =
patientElement.getElementsByTagName("ID").item(0).getTextContent();

                // Módosítjuk a 2 ID-jű patient születési évét, példa: Év
hozzáadása
                if ("2".equals(patientId)) {
                    Element dateOfBirthElement = (Element)
patientElement.getElementsByTagName("DateOfBirth").item(0);
                    LocalDate newDateOfBirth =
LocalDate.parse(dateOfBirthElement.getTextContent()).plusYears(1);
```

```
                    dateOfBirthElement.setTextContent(newDateOfBirth.toString());
                }
            }
        }
    }

    private static void addNewDoctor(Document doc) {
        Element rootElement = doc.getDocumentElement();

        // Új Doctor elem hozzáadása
        Element newDoctorElement = doc.createElement("Doctor");
        Element idElement = doc.createElement("ID");
        idElement.appendChild(doc.createTextNode("105"));
        newDoctorElement.appendChild(idElement);

        Element nameElement = doc.createElement("Name");
        nameElement.appendChild(doc.createTextNode("Dr. New Doctor"));
        newDoctorElement.appendChild(nameElement);

        Element addressElement = doc.createElement("Address");
        addressElement.appendChild(doc.createTextNode("New Doctor Address"));
        newDoctorElement.appendChild(addressElement);

        Element phoneElement = doc.createElement("PhoneNumber");
        phoneElement.appendChild(doc.createTextNode("123456789"));
        newDoctorElement.appendChild(phoneElement);

        Element emailElement = doc.createElement("Email");
        emailElement.appendChild(doc.createTextNode("new.doctor@example.com"));
        newDoctorElement.appendChild(emailElement);

        rootElement.appendChild(newDoctorElement);
    }

    private static void printDocument(Document doc) throws TransformerException {
        TransformerFactory tf = TransformerFactory.newInstance();
        Transformer transformer = tf.newTransformer();
        transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        StringWriter writer = new StringWriter();
        transformer.transform(new DOMSource(doc), new StreamResult(writer));
        String output = writer.getBuffer().toString();
        System.out.println(output);
    }
}
```

h. DOM adatlekérdezés

A DOMQueryTHDWDR osztály egy XML dokumentumot olvas be, majd végrehajt néhány lekérdezést az adatokon. Az a) részben kinyomtatja az orvosok neveit. A b) részben megkeresi a 3-as azonosítójú páciens email címét. A c) részben megtalálja a 2-es azonosítójú páciens születési évét. A d) részben kiírja az 1-es azonosítójú páciens címadatait. Az e) részben megtalálja a 103-as azonosítójú orvos email címét. Végül, az f) részben kinyomtatja, hogy az 1-es azonosítójú páciens mikortól szenved betegségben. A kódban a lekérdezéseket az XML dokumentum elemeinek hierarchiáját felhasználva hajtja végre.

A forráskód:

```java
package hu.domparse.thdwdr;

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

public class DOMQueryTHDWDR {
    public static void main(String[] args) {
      try {
            File inputFile = new File("XMLTHDWDR.xml");
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();

            // a) Orvosok neveinek lekérdezése
            NodeList doctorList = doc.getElementsByTagName("Doctor");
            System.out.println("Orvosok nevei:");
            for (int i = 0; i < doctorList.getLength(); i++) {
                Node node = doctorList.item(i);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element doctor = (Element) node;

System.out.println(doctor.getElementsByTagName("Name").item(0).getTextContent());
                }
            }

            // b) 3 ID-jű patient emailcímének lekérdezése
            String patientIdToQuery = "3";
            NodeList patientList = doc.getElementsByTagName("Patient");
            for (int i = 0; i < patientList.getLength(); i++) {
                Node node = patientList.item(i);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element patient = (Element) node;
                    if
(patient.getElementsByTagName("ID").item(0).getTextContent().equals(patientIdToQue
ry)) {
                        System.out.println("3 ID-jű patient emailcíme: " +

patient.getElementsByTagName("Email").item(0).getTextContent());
                        break;
                    }
                }
            }

            // c) 2 ID-jű patient születési éve
            String patientIdToQueryBirthYear = "2";
            for (int i = 0; i < patientList.getLength(); i++) {
                Node node = patientList.item(i);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element patient = (Element) node;
                    if
(patient.getElementsByTagName("ID").item(0).getTextContent().equals(patientIdToQue
ryBirthYear)) {
                        System.out.println("2 ID-jű patient születési éve: " +

patient.getElementsByTagName("DateOfBirth").item(0).getTextContent());
                        break;
```

```
                    }
                }
            }

            // d) 1 ID-jű patient címadatai
            String patientIdToQueryAddress = "1";
            for (int i = 0; i < patientList.getLength(); i++) {
                Node node = patientList.item(i);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element patient = (Element) node;
                    if
(patient.getElementsByTagName("ID").item(0).getTextContent().equals(patientIdToQue
ryAddress)) {
                        Element address = (Element)
patient.getElementsByTagName("Address").item(0);
                        System.out.println("1 ID-jű patient címadatai:");
                        System.out.println("PostalCode: " +
address.getElementsByTagName("PostalCode").item(0).getTextContent());
                        System.out.println("City: " +
address.getElementsByTagName("City").item(0).getTextContent());
                        System.out.println("Street: " +
address.getElementsByTagName("Street").item(0).getTextContent());
                        System.out.println("HouseNumber: " +
address.getElementsByTagName("HouseNumber").item(0).getTextContent());
                        break;
                    }
                }
            }

            // e) 103 doctor email címe
            String doctorIdToQueryEmail = "103";
            NodeList doctorListForEmail = doc.getElementsByTagName("Doctor");
            for (int i = 0; i < doctorListForEmail.getLength(); i++) {
                Node node = doctorListForEmail.item(i);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element doctor = (Element) node;
                    if
(doctor.getElementsByTagName("ID").item(0).getTextContent().equals(doctorIdToQuery
Email)) {
                        System.out.println("103 ID-jű doctor email címe: " +

doctor.getElementsByTagName("Email").item(0).getTextContent());
                        break;
                    }
                }
            }

            // f) Az 1-es beteg mióta (FromWhen) szenved betegségben?
            String patientIdToQuerySuffering = "1";
            NodeList sufferingList = doc.getElementsByTagName("Suffering");
            for (int i = 0; i < sufferingList.getLength(); i++) {
                Node node = sufferingList.item(i);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element suffering = (Element) node;
                    if
(suffering.getElementsByTagName("PatientID").item(0).getTextContent().equals(patie
ntIdToQuerySuffering)) {
                        System.out.println("Az 1-es beteg mióta szenved
betegségben (FromWhen): " +
```

```
suffering.getElementsByTagName("FromWhen").item(0).getTextContent());
                        break;
                    }
                }
            }

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

i. DOM adatírás

A DOMWriteTHDWDR osztály az XMLTHDWDR.xml fájl beolvasásával kezdi, majd elvégzi néhány módosítást a DOMReadTHDWDR osztály segítségével beolvasott Document objektumon. Az új, módosított dokumentumot elmenti egy új XML fájlba (XMLTHDWDR1.xml), és kiírja az XML struktúrát a konzolra. A saveToXmlFile metódus az átalakított dokumentumot egy új fájlba menti. A printDocument metódus pedig az átalakított dokumentumot írja ki a konzolra. A main metódusban a DOMReadTHDWDR osztály segítségével beolvassák az eredeti XML dokumentumot, majd a módosított dokumentumot elmentik és kiírják a konzolra. A kód XML fájlok kezelésére és módosítására használja a DOM (Document Object Model) API-t.

A forráskód:

```
package hu.domparse.thdwdr;

import org.w3c.dom.Document;

import java.io.File;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

public class DOMWriteTHDWDR {
    public static void main(String[] args) {
        try {
            // Beolvasás a DOMReadTHDWDR osztály segítségével
            Document doc = readXmlDocument("XMLTHDWDR.xml");

            // Mentés az új XML fájlba
            saveToXmlFile(doc, "XMLTHDWDR1.xml");

            // Az XML struktúra kiírása a konzolra
            printDocument(doc);

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

    private static Document readXmlDocument(String fileName) throws Exception {
```

```
        File inputFile = new File(fileName);
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
        return dBuilder.parse(inputFile);
    }
    private static void saveToXmlFile(Document document, String fileName) {
        try {
            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            DOMSource source = new DOMSource(document);
            StreamResult result = new StreamResult(fileName);
            transformer.transform(source, result);
            System.out.println("XML fájl mentve: " + fileName);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void printDocument(Document document) {
        try {
            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
            DOMSource source = new DOMSource(document);
            StreamResult result = new StreamResult(System.out);
            transformer.transform(source, result);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```