JEGYZŐKÖNYV
Adatkezelés XML környezetben

Féléves feladat: Autó kölcsönző

Készítette: **Buha Milán**
Neptunkód: **IY5AM2**
Dátum: **2023. 11. 20.**

Miskolc, 2023

Tartalomjegyzék

1. fejezet

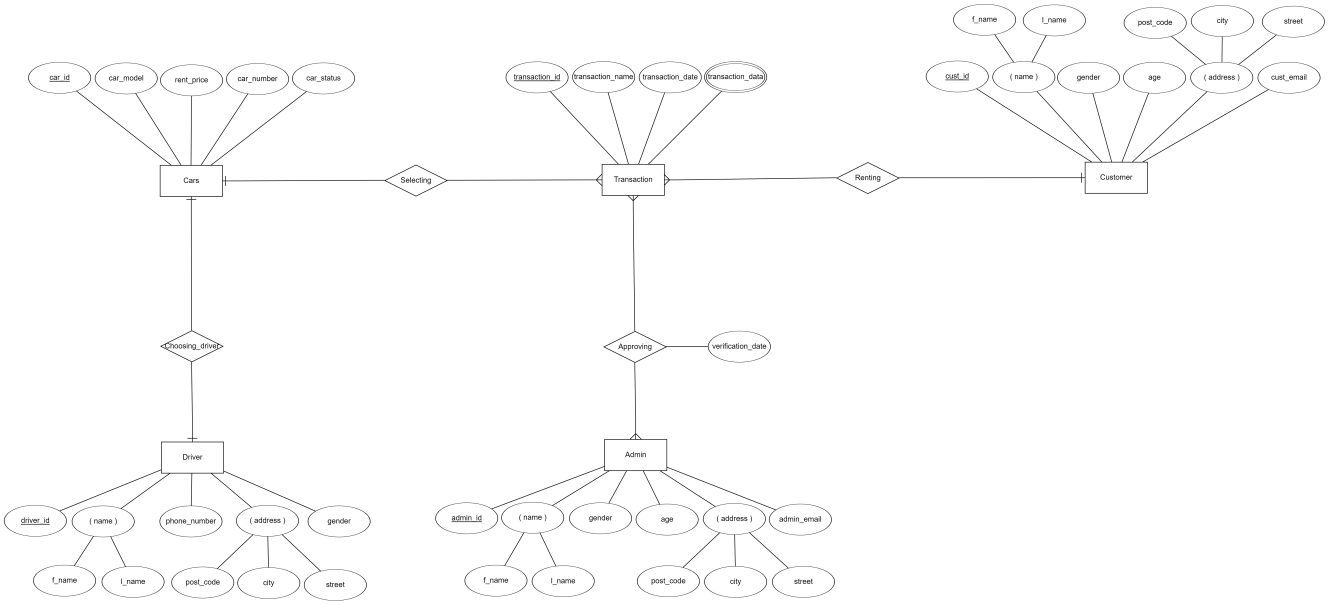
A feladat leírása

A projekt egy autókölcsönző vállalkozás adatait kezeli, amely tartalmazza a különböző tranzakciókat, autókat, sofőröket, ügyfeleket, adminisztrátorokat, és az egyes tranzakciók jóváhagyását. Az adatstruktúra magában foglalja a tranzakciók részleteit, mint a tranzakció nevét és dátumát, valamint a fizetési módokat. Az autók részletei között szerepel a modell, a bérlési díj, a rendszám és az állapot. A sofőrök adatai között szerepel a telefonszám, nem, cím, és a teljes név. Az ügyfelek adatai hasonlóak, beleértve az e-mail címet és az életkort. Az adminisztrátorok adatai tartalmazzák a munkához kapcsolódó e-mail címet, életkort, és címet. A jóváhagyások részletei az adminisztrátorok és a tranzakciók közötti kapcsolatot mutatják be, valamint a jóváhagyás dátumát. Az ER modell és az XML megvalósítás angol nyelven készült, mivel ez a legelterjedtebb nyelv a programozásban.

Összesen 6 egyedet hoztam létre, melyek a következők:

- Transaction,
- Cars,
- Driver,
- Customer,
- Admin,
- Approving

1.1. A feladat ER modellje



1.1. ábra. ER modellje a feladatomnak

Legelőször is érdemes pár szót szólni a **Transaction** entitásról, amely minden tranzakciót kezdeményez. Ez az entitás tárolja az egyes tranzakciók alapvető tulajdonságait, mint például a tranzakció nevét és dátumát. Az elsődleges kulcsa a`transaction_id`, ami a tranzakció azonosítója.

A **Cars** és az **Transaction** egyed között egy `1:N` kapcsolat van, mivel egy tranzakcióhoz több autó is tartozhat, de egy autó csak egy tranzakcióhoz tartozhat. A **Cars** entitásnak van azonosítója, modellje, bérleti díja, rendszáma és állapota. A **Driver**entitás, amely a sofőröket reprezentálja, szintén fontos része a rendszernek. Minden sofőrnek van azonosítója, telefonszáma, neme, címe (ami az ER modellben több részből álló tulajdonság), és teljes neve. Az `1:N` kapcsolat neve: **Renting**. Egy ügyfélnek van azonosítója, neve, neme, életkora, e-mail címe és címe (ami az ER modellben több részből álló tulajdonság), amelyek az ügyfél alapadatait képezik. Fontos, hogy minden ügyfélhez egy vagy több tranzakció tartozik, ami egy többértékű tulajdonságként jelenik meg. Ez a relációs modellnél külön táblát fog kapni, amely tartalmazza a tranzakció azonosítóját, nevét, dátumát és a hozzá kapcsolódó fizetési módot.

Minden autóhoz több sofőr is tartozhat, de egy sofőr csak egy autóhoz tartozik. Ezt ábrázolja a **Drives** kapcsolat, ami `1:N` kapcsolattal köti össze a **Cars** és a **Driver** entitásokat. A sofőrnek van azonosítója, telefonszáma, neme, címe, ami az irányítószámot, várost, utcát és teljes nevet tartalmaz.

Az **Approving** kapcsolat `1:N` kapcsolattal köti össze az **Admin**-t a **Transaction**-nel. Minden adminisztrátornak van azonosítója, neme, életkora, e-mail címe és címe, amelyek az adminisztrátor

alapadatait képezik.

Az **Transaction**-t és a **Customer**-t összekötő `1:1` kapcsolatot a **Renting** képviseli. A **Transaction** entitás tartalmazza a tranzakció azonosítóját, nevét, dátumát és a fizetési módját, míg a **Customer** az ügyfél adatait képviseli.

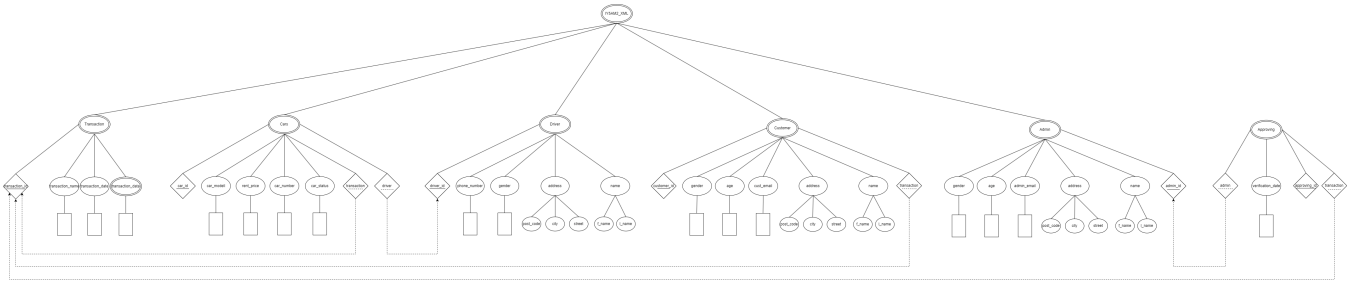
Minden **Transaction** entitáshoz tartozik egy **Approving** entitás, ami az adminisztrátor által történő jóváhagyást képviseli. Ez tartalmazza a jóváhagyás azonosítóját, az adminisztrátor azonosítóját, a tranzakció azonosítóját és a jóváhagyás dátumát.

2. fejezet

XML/XSD létrehozás

2.1. A feladat XDM modellje

A konvertáláskor figyelembe kell venni az ER modell során definiált kapcsolatokat, azok típusait (`1:1, 1:N, N:M`), illetve az entitások elsődleges kulcsait is. Minden *egy-több* kapcsolat esetében ahhoz az elsődleges kulcshoz kerül a szaggatott nyíl, ahol az ER modellben a „több" szerepel. Az „Admin" és a „Driver" kapcsolatokat kivéve, mindenhol `1:N` kapcsolat szerepel az ER modellben, így az XDM mindenhol majdnem hasonlóan fog kinézni. Az `N:M` kapcsolat esetében egy új modellt veszünk fel, tulajdonsággal és *primary key*-el együtt természetesen, ahonnan a nyilakat a fő entitásokhoz húzzuk. A többágú tulajdonságok itt is több tulajdonsággal rendelkeznek, a többértékű tulajdonságok itt nem kapnak külön modellt. Az XDM modell gyökéreleme: **Car_IY5AM2**



2.1. ábra. XDM modellje a feladatomnak

A „Renting" kapcsolat `1:N`, ahol a több érték a *Customer*-hez kerül, így a kapcsolatot is a *Customer*-hez húzzuk. Szintén ugyan ez a helyzet a „Choosing_driver" kapcsolatnál is, ahol a rombuszt a *driver_id*-hez húzzuk. Az *Transaction* modell egy különleges, ide több kapcsolatot is húzunk, melyek:

- „Choosing_driver" a *Driver*-ból

- „Renting” a *Customer*-ből és végül
- „Approving” az *Admin*-ből

Két `1:N` kapcsolat és egy `N:M` kapcsolat húz ide.

2.2. Az XML modell alapján XML dokumentum készítése

Az `Car_IY5AM2.xml` dokumentumot *Visual Studio Code*-ban hoztam létre, és `XML 1.0` szabvány szerint készült el. A dokumentumhoz hozzá kötöttem az `IY5AM2XSD.xsd` XSD file-t, és definiáltam az egyedeket az XML szabályainak megfelelően. Ahol szükséges volt, gyermek elemeket, valamint attribútumokat használtam a tagok azonosításához.

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

<Car_IY5AM2 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="IY5AM2XSD.xsd">

    <Transaction transaction_id="1">
        <transaction_name>Tesomnak</transaction_name>
        <transaction_date>2023-11-14</transaction_date>
        <transactions>
            <transaction_data>Cash</transaction_data>
        </transactions>
    </Transaction>

    <Transaction transaction_id="2">
        <transaction_name>Uzleti</transaction_name>
        <transaction_date>2023-11-15</transaction_date>
        <transactions>
            <transaction_data>Bank Transfer</transaction_data>
        </transactions>
    </Transaction>

    <Transaction transaction_id="3">
        <transaction_name>Szuletesnapi Ajandek</transaction_name>
        <transaction_date>2023-11-16</transaction_date>
        <transactions>
            <transaction_data>Credit Card</transaction_data>
        </transactions>
    </Transaction>

    <Cars car_id="1" transaction="1" driver="1">
        <car_modell>Opel</car_modell>
        <rent_price>200.0</rent_price>
        <car_number>69</car_number>
```

```
    <car_status>Available</car_status>
</Cars>

<Cars car_id="3" transaction="2" driver="3">
    <car_modell>Toyota</car_modell>
    <rent_price>150.0</rent_price>
    <car_number>33</car_number>
    <car_status>Unavailable</car_status>
</Cars>

<Cars car_id="4" transaction="3" driver="2">
    <car_modell>BMW</car_modell>
    <rent_price>300.0</rent_price>
    <car_number>44</car_number>
    <car_status>Available</car_status>
</Cars>

<Driver driver_id="1">
    <phone_number>06501311010</phone_number>
    <gender>Male</gender>
    <post_code>3534</post_code>
    <city>Miskolc</city>
    <street>Arpad</street>
    <first_name>Buha</first_name>
    <last_name>Milan</last_name>
</Driver>

<Driver driver_id="2">
    <phone_number>06502223344</phone_number>
    <gender>Male</gender>
    <post_code>1122</post_code>
    <city>Debrecen</city>
    <street>Kossuth</street>
    <first_name>Szabo</first_name>
    <last_name>Laszlo</last_name>
</Driver>

<Driver driver_id="3">
    <phone_number>06505556677</phone_number>
    <gender>Female</gender>
    <post_code>2045</post_code>
    <city>Torokbalint</city>
    <street>Fo</street>
    <first_name>Nagy</first_name>
    <last_name>Anna</last_name>
</Driver>

<Customer customer_id="1" transaction="1">
```

```
        <gender>Female</gender>
        <age>20</age>
        <cust_email>barbara@gmail.com</cust_email>
        <post_code>3535</post_code>
        <city>Miskolc</city>
        <street>Kuruc</street>
        <first_name>Valaki</first_name>
        <last_name>Barbara</last_name>
    </Customer>

    <Customer customer_id="2" transaction="2">
        <gender>Male</gender>
        <age>30</age>
        <cust_email>istvan@gmail.com</cust_email>
        <post_code>6000</post_code>
        <city>Kecskemet</city>
        <street>Petofi</street>
        <first_name>Istvan</first_name>
        <last_name>Szabo</last_name>
    </Customer>

    <Customer customer_id="3" transaction="3">
        <gender>Female</gender>
        <age>25</age>
        <cust_email>eszter@gmail.com</cust_email>
        <post_code>6722</post_code>
        <city>Szeged</city>
        <street>Jozsef</street>
        <first_name>Eszter</first_name>
        <last_name>Kovacs</last_name>
    </Customer>

    <Admin admin_id="1">
        <gender>Male</gender>
        <age>32</age>
        <admin_email>admin@gmail.com</admin_email>
        <post_code>3634</post_code>
        <city>Valahol</city>
        <street>Napfeny</street>
        <first_name>Kiss</first_name>
        <last_name>Janos</last_name>
    </Admin>

    <Admin admin_id="2">
        <gender>Female</gender>
        <age>28</age>
        <admin_email>admin2@gmail.com</admin_email>
        <post_code>7632</post_code>
```

```
        <city>Pecs</city>
        <street>Hunyadi</street>
        <first_name>Toth</first_name>
        <last_name>Krisztina</last_name>
    </Admin>

    <Admin admin_id="3">
        <gender>Male</gender>
        <age>40</age>
        <admin_email>admin3@gmail.com</admin_email>
        <post_code>1012</post_code>
        <city>Budapest</city>
        <street>Var</street>
        <first_name>Varga</first_name>
        <last_name>Mihaly</last_name>
    </Admin>

    <Approving approving_id="1" admin="1" transaction="1">
        <verification_date>2023-11-14</verification_date>
    </Approving>

    <Approving approving_id="2" admin="2" transaction="2">
        <verification_date>2023-11-15</verification_date>
    </Approving>

    <Approving approving_id="3" admin="3" transaction="3">
        <verification_date>2023-11-16</verification_date>
    </Approving>

</Car_IY5AM2>
```

Programkód 2.1. XML dokumentum

2.3. Az XML dokumentum alapján XMLSchema készítése

Az `XMLSchemaIY5AM2.xsd` séma file leírja mindazon megkötéseket, amelyeknek az XML dokumentumnak meg kell felelnie. Itt definiálunk minden típust, amit az XML file-ban használni szeretnénk, valamint az adatbázis kapcsolatait `xs:unique` és `xs:keyref` bejegyzésekkel hozom létre.

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

    <!-- Sajat tipusok -->
    <xs:element name="phone_number" type="xs:string"/>
    <xs:element name="gender" type="xs:string"/>
    <xs:element name="post_code" type="xs:string"></xs:element>
    <xs:element name="city" type="xs:string"></xs:element>
    <xs:element name="street" type="xs:string"></xs:element>
    <xs:element name="first_name" type="xs:string"></xs:element>
    <xs:element name="last_name" type="xs:string"></xs:element>

    <xs:element name="transaction_name" type="xs:string"/>
    <xs:element name="transaction_date" type="xs:date"/>

    <xs:element name="car_modell" type="xs:string"/>
    <xs:element name="rent_price" type="xs:decimal"/>
    <xs:element name="car_number" type="xs:string"/>
    <xs:element name="car_status" type="StatusType"/>

    <xs:element name="age" type="xs:integer"/>
    <xs:element name="cust_email" type="xs:string"/>
    <xs:element name="admin_email" type="xs:string"/>

    <xs:element name="transactions" type="TransactionDataType"></xs:element>

    <!-- TransactionType Simple Type -->
    <xs:simpleType name="TransactionDataAttributeType">
        <xs:restriction base="xs:string">
        </xs:restriction>
    </xs:simpleType>

    <!-- GenderType Simple Type -->
    <xs:simpleType name="GenderType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Male"/>
            <xs:enumeration value="Female"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- StatusType Simple Type -->
```

```
<xs:simpleType name="StatusType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Available"/>
        <xs:enumeration value="Unavailable"/>
    </xs:restriction>
</xs:simpleType>

<!-- Root Complex Type -->
<xs:complexType name="CarType">
    <xs:sequence>
        <xs:element name="Transaction" type="TransactionType"
minOccurs="3" maxOccurs="unbounded"/>
        <xs:element name="Cars" type="CarsType" minOccurs="3"
maxOccurs="unbounded"/>
        <xs:element name="Driver" type="DriverType" minOccurs="3"
maxOccurs="unbounded"/>
        <xs:element name="Customer" type="CustomerType" minOccurs="3"
maxOccurs="unbounded"/>
        <xs:element name="Admin" type="AdminType" minOccurs="3"
maxOccurs="unbounded"/>
        <xs:element name="Approving" type="ApprovingType" minOccurs="3"
maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>

<!-- TransactionType Complex Type -->
<xs:complexType name="TransactionType">
    <xs:sequence>
        <xs:element ref="transaction_name"/>
        <xs:element ref="transaction_date"/>
        <xs:element ref="transactions"/>
    </xs:sequence>
    <xs:attribute name="transaction_id" type="xs:integer" use="required"/>
</xs:complexType>

<xs:complexType name="TransactionDataType">
    <xs:sequence>
        <xs:element name="transaction_data" type="xs:string" minOccurs="1"
maxOccurs="unbounded"></xs:element>
    </xs:sequence>
    <xs:attribute name="transaction_data"
type="TransactionDataAttributeType"/>
</xs:complexType>

<!-- CarsType Complex Type -->
<xs:complexType name="CarsType">
    <xs:sequence>
        <xs:element ref="car_modell"/>
```

```
            <xs:element ref="rent_price"/>
            <xs:element ref="car_number"/>
            <xs:element ref="car_status"/>
        </xs:sequence>
        <xs:attribute name="car_id" type="xs:integer" use="required"/>
        <xs:attribute name="transaction" type="xs:integer" use="required"/>
        <xs:attribute name="driver" type="xs:integer" use="required"/>
    </xs:complexType>

    <!-- DriverType Complex Type -->
    <xs:complexType name="DriverType">
        <xs:sequence>
            <xs:element ref="phone_number"/>
            <xs:element ref="gender"/>
            <xs:element ref="post_code"/>
            <xs:element ref="city"/>
            <xs:element ref="street"/>
            <xs:element ref="first_name"/>
            <xs:element ref="last_name"/>
        </xs:sequence>
        <xs:attribute name="driver_id" type="xs:integer" use="required"/>
    </xs:complexType>

    <!-- CustomerType Complex Type -->
    <xs:complexType name="CustomerType">
        <xs:sequence>
            <xs:element ref="gender"/>
            <xs:element ref="age"/>
            <xs:element ref="cust_email"/>
            <xs:element ref="post_code"/>
            <xs:element ref="city"/>
            <xs:element ref="street"/>
            <xs:element ref="first_name"/>
            <xs:element ref="last_name"/>
        </xs:sequence>
        <xs:attribute name="customer_id" type="xs:integer" use="required"/>
        <xs:attribute name="transaction" type="xs:integer" use="required"/>
    </xs:complexType>

    <!-- AdminType Complex Type -->
    <xs:complexType name="AdminType">
        <xs:sequence>
            <xs:element ref="gender"/>
            <xs:element ref="age"/>
            <xs:element ref="admin_email"/>
            <xs:element ref="post_code"/>
            <xs:element ref="city"/>
            <xs:element ref="street"/>
```

```
            <xs:element ref="first_name"/>
            <xs:element ref="last_name"/>
        </xs:sequence>
        <xs:attribute name="admin_id" type="xs:integer" use="required"/>
    </xs:complexType>

    <!-- ApprovingType Complex Type -->
    <xs:complexType name="ApprovingType">
        <xs:sequence>
            <xs:element name="verification_date" type="xs:date"/>
        </xs:sequence>
        <xs:attribute name="approving_id" type="xs:integer" use="required"/>
        <xs:attribute name="admin" type="xs:integer" use="required"/>
        <xs:attribute name="transaction" type="xs:integer" use="required"/>
    </xs:complexType>

    <xs:element name="Car_IY5AM2" type="CarType">

        <xs:key name="transactionKey">
            <xs:selector xpath="Transaction"/>
            <xs:field xpath="@transaction_id"/>
        </xs:key>

        <xs:key name="driverKey">
            <xs:selector xpath="Driver"/>
            <xs:field xpath="@driver_id"/>
        </xs:key>

        <xs:key name="carsKey">
            <xs:selector xpath="Cars"/>
            <xs:field xpath="@car_id"/>
        </xs:key>

        <xs:key name="customerKey">
            <xs:selector xpath="Customer"/>
            <xs:field xpath="@customer_id"/>
        </xs:key>

        <xs:key name="adminKey">
            <xs:selector xpath="Admin"/>
            <xs:field xpath="@admin_id"/>
        </xs:key>

        <xs:key name="approvingKey">
            <xs:selector xpath="Approving"/>
            <xs:field xpath="@approving_id"/>
        </xs:key>
```

```
        <xs:keyref name="carTransaction" refer="transactionKey">
            <xs:selector xpath="Cars"/>
            <xs:field xpath="@transaction"/>
        </xs:keyref>

        <xs:keyref name="carDriver" refer="driverKey">
            <xs:selector xpath="Cars"/>
            <xs:field xpath="@transaction"/>
        </xs:keyref>

        <xs:keyref name="customerTransaction" refer="transactionKey">
            <xs:selector xpath="Customer"/>
            <xs:field xpath="@transaction"/>
        </xs:keyref>

        <xs:keyref name="approvingAdmin" refer="adminKey">
            <xs:selector xpath="Approving"/>
            <xs:field xpath="@admin"/>
        </xs:keyref>

        <xs:keyref name="approvingTransaction" refer="transactionKey">
            <xs:selector xpath="Approving"/>
            <xs:field xpath="@transaction"/>
        </xs:keyref>

        <!-- 1:1 -->
        <xs:unique name="DriverCarConnect">
            <xs:selector xpath="carKey"/>
            <xs:field xpath="@driver"/>
        </xs:unique>

    </xs:element>

</xs:schema>
```

Programkód 2.2. XSD dokumentum

3. fejezet

DOM

3.1. Adatolvasás

Ez a Java program, a `DOMReadIY5AM2`, egy XML fájlt olvas be és dolgoz fel a DOM (Document Object Model) parser segítségével. A DOM parser a teljes XML dokumentumot memóriába tölti, ami gyors hozzáférést biztosít az elemekhez, de nagyobb dokumentumok esetén jelentős memóriaigényt jelenthet. A program beolvassa az XML fájlt, normalizálja azt, és különböző függvények segítségével feldolgozza az XML elemeket, melyek az 'Transactions', 'Cars', 'Drivers', 'Customers', 'Admins' és 'Approvings'. Minden elemcsoport feldolgozása külön függvényben történik, ami javítja a kód olvashatóságát és karbantarthatóságát. Hibakezelés is implementálva van a fájlbeolvasás és parse-lás során.

```
package domparse.IY5AM2;

import javax.xml.parsers.*;

import org.xml.sax.SAXException;
import org.w3c.dom.*;

import java.io.*;

public class DOMReadIY5AM2 {

    public static void main(String[] args) {
        try {
            DocumentBuilderFactory factory =
   DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse(new
   File("C:\\Users\\buha3\\Desktop\\XMLTaskIY5AM2\\IY5AM2XML.xml"));

            document.getDocumentElement().normalize();
```

```
            System.out.println("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");
            System.out.println("<Car_IY5AM2
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation=\"IY5AM2XSD.xsd\">\n");

            // Elemeket beolvas  met dusok
            readTransactions(document);
            readCars(document);
            readDrivers(document);
            readCustomers(document);
            readAdmins(document);
            readApprovings(document);

            System.out.println("\n</Car_IY5AM2>");
        } catch (ParserConfigurationException | IOException | SAXException e) {
            System.out.println("Error: " + e);
        }
    }

    // Transaction Node beolvas  met dus
    private static void readTransactions(Document document) {
        NodeList transactionList =
document.getElementsByTagName("Transaction");
        for (int temp = 0; temp < transactionList.getLength(); temp++) {
            Node node = transactionList.item(temp);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) node;
                String transactionId = eElement.getAttribute("transaction_id");
                String transactionName =
eElement.getElementsByTagName("transaction_name").item(0).getTextContent();
                String transactionDate =
eElement.getElementsByTagName("transaction_date").item(0).getTextContent();

                System.out.println("    <Transaction transaction_id=\"" +
transactionId + "\">");
                printElement("transaction_name", transactionName);
                printElement("transaction_date", transactionDate);

                // Transaction Data elemek kiolvas sa
                NodeList transactionDataList =
eElement.getElementsByTagName("transaction_data");
                System.out.println("        <Transactions>");
                for (int i = 0; i < transactionDataList.getLength(); i++) {
                    String transactionData =
transactionDataList.item(i).getTextContent();
                    System.out.println("            <transaction_data>" +
transactionData + "</transaction_data>");
                }
```

```
            System.out.println("        </Transactions>");

            System.out.println("    </Transaction>");
        }
    }
}

// Cars Node beolvas  met dus
private static void readCars(Document document) {
    NodeList carsList = document.getElementsByTagName("Cars");
    for (int temp = 0; temp < carsList.getLength(); temp++) {
        Node node = carsList.item(temp);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            String carId = eElement.getAttribute("car_id");
            String carModel =
eElement.getElementsByTagName("car_modell").item(0).getTextContent();
            String rentPrice =
eElement.getElementsByTagName("rent_price").item(0).getTextContent();
            String carNumber =
eElement.getElementsByTagName("car_number").item(0).getTextContent();
            String carStatus =
eElement.getElementsByTagName("car_status").item(0).getTextContent();

            System.out.println("    <Cars car_id=\"" + carId + "\">");
            printElement("car_modell", carModel);
            printElement("rent_price", rentPrice);
            printElement("car_number", carNumber);
            printElement("car_status", carStatus);
            System.out.println("    </Cars>");
        }
    }
}

// Driver Node beolvas  met dus
private static void readDrivers(Document document) {
    NodeList driverList = document.getElementsByTagName("Driver");
    for (int temp = 0; temp < driverList.getLength(); temp++) {
        Node node = driverList.item(temp);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            String driverId = eElement.getAttribute("driver_id");
            String phoneNumber =
eElement.getElementsByTagName("phone_number").item(0).getTextContent();
            String gender =
eElement.getElementsByTagName("gender").item(0).getTextContent();
            String postCode =
eElement.getElementsByTagName("post_code").item(0).getTextContent();
```

```
            String city =
eElement.getElementsByTagName("city").item(0).getTextContent();
            String street =
eElement.getElementsByTagName("street").item(0).getTextContent();
            String firstName =
eElement.getElementsByTagName("first_name").item(0).getTextContent();
            String lastName =
eElement.getElementsByTagName("last_name").item(0).getTextContent();

            System.out.println("    <Driver driver_id=\"" + driverId +
"\">");
            printElement("phone_number", phoneNumber);
            printElement("gender", gender);
            printElement("post_code", postCode);
            printElement("city", city);
            printElement("street", street);
            printElement("first_name", firstName);
            printElement("last_name", lastName);
            System.out.println("    </Driver>");
        }
    }
}

// Customer Node beolvas  met dus
private static void readCustomers(Document document) {
    NodeList customerList = document.getElementsByTagName("Customer");
    for (int temp = 0; temp < customerList.getLength(); temp++) {
        Node node = customerList.item(temp);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            String customerId = eElement.getAttribute("customer_id");
            String gender =
eElement.getElementsByTagName("gender").item(0).getTextContent();
            String age =
eElement.getElementsByTagName("age").item(0).getTextContent();
            String email =
eElement.getElementsByTagName("cust_email").item(0).getTextContent();
            String postCode =
eElement.getElementsByTagName("post_code").item(0).getTextContent();
            String city =
eElement.getElementsByTagName("city").item(0).getTextContent();
            String street =
eElement.getElementsByTagName("street").item(0).getTextContent();
            String firstName =
eElement.getElementsByTagName("first_name").item(0).getTextContent();
            String lastName =
eElement.getElementsByTagName("last_name").item(0).getTextContent();
```

```
            System.out.println("    <Customer customer_id=\"" + customerId
+ "\">");
            printElement("gender", gender);
            printElement("age", age);
            printElement("cust_email", email);
            printElement("post_code", postCode);
            printElement("city", city);
            printElement("street", street);
            printElement("first_name", firstName);
            printElement("last_name", lastName);
            System.out.println("    </Customer>");
        }
    }
}

// Admin Node beolvas  met dus
private static void readAdmins(Document document) {
    NodeList adminList = document.getElementsByTagName("Admin");
    for (int temp = 0; temp < adminList.getLength(); temp++) {
        Node node = adminList.item(temp);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element eElement = (Element) node;
            String adminId = eElement.getAttribute("admin_id");
            String gender =
eElement.getElementsByTagName("gender").item(0).getTextContent();
            String age =
eElement.getElementsByTagName("age").item(0).getTextContent();
            String email =
eElement.getElementsByTagName("admin_email").item(0).getTextContent();
            String postCode =
eElement.getElementsByTagName("post_code").item(0).getTextContent();
            String city =
eElement.getElementsByTagName("city").item(0).getTextContent();
            String street =
eElement.getElementsByTagName("street").item(0).getTextContent();
            String firstName =
eElement.getElementsByTagName("first_name").item(0).getTextContent();
            String lastName =
eElement.getElementsByTagName("last_name").item(0).getTextContent();

            System.out.println("    <Admin admin_id=\"" + adminId + "\">");
            printElement("gender", gender);
            printElement("age", age);
            printElement("admin_email", email);
            printElement("post_code", postCode);
            printElement("city", city);
            printElement("street", street);
            printElement("first_name", firstName);
```

```
                printElement("last_name", lastName);
                System.out.println("    </Admin>");
            }
        }
    }

    // Approving Node beolvas  met dus
    private static void readApprovings(Document document) {
        NodeList approvingList = document.getElementsByTagName("Approving");
        for (int temp = 0; temp < approvingList.getLength(); temp++) {
            Node node = approvingList.item(temp);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element eElement = (Element) node;
                String approvingId = eElement.getAttribute("approving_id");
                String admin = eElement.getAttribute("admin");
                String transaction = eElement.getAttribute("transaction");
                String verificationDate =
   eElement.getElementsByTagName("verification_date").item(0).getTextContent();

                System.out.println("    <Approving approving_id=\"" +
   approvingId + "\" admin=\"" + admin + "\" transaction=\"" + transaction +
   "\">");
                printElement("verification_date", verificationDate);
                System.out.println("    </Approving>");
            }
        }
    }

    // Elem ki rat  met dus
    private static void printElement(String elementName, String content) {
        System.out.println("        <" + elementName + ">" + content + "</" +
   elementName + ">");
    }
}
```

Programkód 3.1. DOMReadIY5AM2.java adatolvasó program

3.2. Adatmódosítás

Ez a Java program, a `DOMModifyIY5AM2`, egy XML fájlt olvas be és módosítja azt a DOM (Document Object Model) API segítségével. Az XML fájl, amely egy autókölcsönző vállalat adatait tartalmazza, egy előre meghatározott útvonalon található, és a `DocumentBuilder` osztály segítségével parse-oljuk. A program három fő részre oszlik: `modifyDrivers`, `modifyCars`, és `modifyCustomers` metódusokra, melyek különböző XML elemeket módosítanak. Az `modifyDrivers` metódus a Driver elemek `phone_number` elemét módosítja, minden `phone_number` elé ,,`MODIFIED_`'' előtagot illeszt-

ve. A `modifyCars` metódus a Cars elemek `car_status` elemét állítja be „Unavailable”-re, ha az eredeti státusz „Available”. A `modifyCustomers` metódus a Customer elemek `age` elemének értékét növeli öttel. A módosítások után a program egy `Transformer` segítségével visszaalakítja és kiírja a módosított DOM-ot XML formátumban. Az XML kiírás során a `Transformer` beállításai biztosítják a formázott, olvasható kimenetet, az `OutputKeys.INDENT` beállítás segítségével.

```
package domparse.IY5AM2;

import javax.xml.parsers.*;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.*;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

public class DOMModifyIY5AM2 {

    public static void main(String[] argv) {
        try {
            File inputFile = new
   File("C:\\Users\\buha3\\Desktop\\XMLTaskIY5AM2\\IY5AM2XML.xml");

            DocumentBuilderFactory docFactory =
   DocumentBuilderFactory.newInstance();
            DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
            Document doc = docBuilder.parse(inputFile);

            // List ba helyezz k a k v nt telefonsz mokat
            List<String> phoneNumbers = new ArrayList<>();
            phoneNumbers.add("123456789");
            phoneNumbers.add("987654321");
            phoneNumbers.add("555555555");

            // M dos t  met dusok megh v sa
            modifyDrivers(doc, phoneNumbers);
            modifyCars(doc);
            modifyCustomers(doc);

            TransformerFactory transformerFactory =
   TransformerFactory.newInstance();
            Transformer transformer = transformerFactory.newTransformer();
```

```
            transformer.setOutputProperty(OutputKeys.INDENT, "no");
            DOMSource source = new DOMSource(doc);
            StreamResult consoleResult = new StreamResult(System.out);
            transformer.transform(source, consoleResult);
        } catch (Exception e) {
            System.out.println("Error: " + e);
        }
    }

    // Driver elemeket m dos t   met dus
    private static void modifyDrivers(Document doc, List<String> phoneNumbers)
   {
        NodeList driverList = doc.getElementsByTagName("Driver");
        int phoneNumberIndex = 0;

        for (int i = 0; i < driverList.getLength(); i++) {
            Node driver = driverList.item(i);
            Element eElement = (Element) driver;

            // Ellen rizz k, hogy m g van-e telefonsz m a list ban
            if (phoneNumberIndex < phoneNumbers.size()) {
                String newPhoneNumber = phoneNumbers.get(phoneNumberIndex);

   eElement.getElementsByTagName("phone_number").item(0).setTextContent(newPhoneNumber);
                phoneNumberIndex++;
            }
        }
    }

    // Cars elemeket m dos t   met dus
    private static void modifyCars(Document doc) {
        NodeList carsList = doc.getElementsByTagName("Cars");
        for (int i = 0; i < carsList.getLength(); i++) {
            Node car = carsList.item(i);
            Element eElement = (Element) car;
            if
   ("Available".equals(eElement.getElementsByTagName("car_status").item(0).getTextContent
   {

   eElement.getElementsByTagName("car_status").item(0).setTextContent("Unavailable");
            }
        }
    }

    // Customer elemeket m dos t   met dus
    private static void modifyCustomers(Document doc) {
        NodeList customerList = doc.getElementsByTagName("Customer");
        for (int i = 0; i < customerList.getLength(); i++) {
```

```
            Node customer = customerList.item(i);
            Element eElement = (Element) customer;
            int age =
    Integer.parseInt(eElement.getElementsByTagName("age").item(0).getTextContent());

    eElement.getElementsByTagName("age").item(0).setTextContent(String.valueOf(age
    + 5));
         }
     }
}
```

Programkód 3.2. DOMQModifyIY5AM2.java adatmódosító program

3.3. Adatlekérdezés

A program a Java `DOM Parser`-t használja XML fájlunk feldolgozására, ami lehetővé teszi a XML elemek olvasását és manipulálását egy objektumorientált módon. A kód, az XML fájlt, a `File` objektumon keresztül tölti be, biztosítva ezzel a fájl elérését és kezelését. A `DocumentBuilderFactory` és `DocumentBuilder` osztályok használata a fájl DOM reprezentációjának létrehozásához szükséges, ami egy strukturált, fa-szerű modellt biztosít az XML adatok számára. A program minden egyes lekérdezést egy for ciklus segítségével hajt végre, ahol a `getElementsByTagName` metódus segítségével specifikus XML elemeket keres. Az elemek feldolgozása során a `Node` és `Element` interfészeket használja, amelyek lehetővé teszik az egyes elemek attribútumainak és tartalmának elérését. A lekérdezések eredményét egy `StringBuilder` objektumba gyűjti, amely hatékonyan kezeli a nagy mennyiségű stringek összefűzését. A kód, az összegyűjtött adatokat XML-szerű formátumban állítja elő.

```
package domparse.IY5AM2;

import java.io.*;
import javax.xml.parsers.*;

import org.w3c.dom.*;
import org.xml.sax.SAXException;

public class DOMQueryIY5AM2 {

    public static void main(String[] argv) throws SAXException, IOException,
    ParserConfigurationException {
        File xmlFile = new
    File("C:\\Users\\buha3\\Desktop\\XMLTaskIY5AM2\\IY5AM2XML.xml");

        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilder = factory.newDocumentBuilder();
```

```
        Document doc = dBuilder.parse(xmlFile);
        doc.getDocumentElement().normalize();

        StringBuilder outputBuilder = new StringBuilder();

        // Lekrdezs a 'Toyota' modellel rendelkez aut kra
        NodeList carsList = doc.getElementsByTagName("Cars");
        outputBuilder.append("\n<ToyotaCars>\n");
        for (int i = 0; i < carsList.getLength(); i++) {
            Node node = carsList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;
                String model =
    element.getElementsByTagName("car_modell").item(0).getTextContent();
                if (model.equals("Toyota")) {
                    String carId = element.getAttribute("car_id");
                    String rentPrice =
    element.getElementsByTagName("rent_price").item(0).getTextContent();
                    outputBuilder.append(String.format("  <Car
    car_id=\"%s\">\n", carId));
                    outputBuilder.append(String.format("
    <Model>%s</Model>\n", model));
                    outputBuilder.append(String.format("
    <RentPrice>%s</RentPrice>\n", rentPrice));
                    outputBuilder.append("  </Car>\n");
                }
            }
        }
        outputBuilder.append("</ToyotaCars>\n");

        // Lekrdez s a 'Male' nem  sof r kre
        NodeList driverList = doc.getElementsByTagName("Driver");
        outputBuilder.append("\n<MaleDrivers>\n");
        for (int i = 0; i < driverList.getLength(); i++) {
            Node node = driverList.item(i);
            if (node.getNodeType() == Node.ELEMENT_NODE) {
                Element element = (Element) node;
                String gender =
    element.getElementsByTagName("gender").item(0).getTextContent();
                if (gender.equals("Male")) {
                    String driverId = element.getAttribute("driver_id");
                    String firstName =
    element.getElementsByTagName("first_name").item(0).getTextContent();
                    String lastName =
    element.getElementsByTagName("last_name").item(0).getTextContent();
                    outputBuilder.append(String.format("  <Driver
    driver_id=\"%s\">\n", driverId));
```

```
                outputBuilder.append(String.format("
<FirstName>%s</FirstName>\n", firstName));
                outputBuilder.append(String.format("
<LastName>%s</LastName>\n", lastName));
                outputBuilder.append(String.format("
<Gender>%s</Gender>\n", gender));
                outputBuilder.append("  </Driver>\n");
            }
        }
    }
    outputBuilder.append("</MaleDrivers>\n");

    // Lek rdez s a 25   vnl   fiatalabb   gyfelekre
    NodeList customerList = doc.getElementsByTagName("Customer");
    outputBuilder.append("\n<YoungCustomers>\n");
    for (int i = 0; i < customerList.getLength(); i++) {
        Node node = customerList.item(i);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;
            int age =
Integer.parseInt(element.getElementsByTagName("age").item(0).getTextContent());
            if (age < 25) {
                String customerId = element.getAttribute("customer_id");
                String firstName =
element.getElementsByTagName("first_name").item(0).getTextContent();
                String lastName =
element.getElementsByTagName("last_name").item(0).getTextContent();
                outputBuilder.append(String.format("  <Customer
customer_id=\"%s\">\n", customerId));
                outputBuilder.append(String.format("
<FirstName>%s</FirstName>\n", firstName));
                outputBuilder.append(String.format("
<LastName>%s</LastName>\n", lastName));
                outputBuilder.append(String.format("    <Age>%d</Age>\n",
age));
                outputBuilder.append("  </Customer>\n");
            }
        }
    }
    outputBuilder.append("</YoungCustomers>\n");

    // Lek rdez s az adminisztr torokra Budapest v ros b l
    NodeList adminList = doc.getElementsByTagName("Admin");
    outputBuilder.append("\n<BudapestAdmins>\n");
    for (int i = 0; i < adminList.getLength(); i++) {
        Node node = adminList.item(i);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
            Element element = (Element) node;
```

```
            String city =
element.getElementsByTagName("city").item(0).getTextContent();
            if (city.equals("Budapest")) {
                String adminId = element.getAttribute("admin_id");
                String firstName =
element.getElementsByTagName("first_name").item(0).getTextContent();
                String lastName =
element.getElementsByTagName("last_name").item(0).getTextContent();
                outputBuilder.append(String.format("  <Admin
admin_id=\"%s\">\n", adminId));
                outputBuilder.append(String.format("
<FirstName>%s</FirstName>\n", firstName));
                outputBuilder.append(String.format("
<LastName>%s</LastName>\n", lastName));
                outputBuilder.append(String.format("
<City>%s</City>\n", city));
                outputBuilder.append("  </Admin>\n");
            }
        }
    }
    outputBuilder.append("</BudapestAdmins>\n");

    // Lek rdez s azokra a tranzakci kra, ahol a b rl s  ra
meghaladja az  tlagos  b rl si d jat
    NodeList transactionList = doc.getElementsByTagName("Transaction");
    double totalRent = 0;
    int transactionCount = 0;

    // Sz m tsuk ki az  tlagos  b rl si d jat
    for (int i = 0; i < transactionList.getLength(); i++) {
        Node transactionNode = transactionList.item(i);
        if (transactionNode.getNodeType() == Node.ELEMENT_NODE) {
            Element transactionElement = (Element) transactionNode;
            String transactionId =
transactionElement.getAttribute("transaction_id");

            // Az aut k adatainak keres se a tranzakci  azonos t
alapj n
            for (int j = 0; j < carsList.getLength(); j++) {
                Node carNode = carsList.item(j);
                if (carNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element carElement = (Element) carNode;
                    if
(carElement.getAttribute("transaction").equals(transactionId)) {
                        double rentPrice =
Double.parseDouble(carElement.getElementsByTagName("rent_price").item(0).getTextConter
                        totalRent += rentPrice;
                        transactionCount++;
```

```
                    }
                }
            }
        }
    }

    double averageRent = totalRent / transactionCount;

    outputBuilder.append("\n<TransactionsAboveAverageRent>\n");
    for (int i = 0; i < transactionList.getLength(); i++) {
        Node transactionNode = transactionList.item(i);
        if (transactionNode.getNodeType() == Node.ELEMENT_NODE) {
            Element transactionElement = (Element) transactionNode;
            String transactionId =
transactionElement.getAttribute("transaction_id");

            for (int j = 0; j < carsList.getLength(); j++) {
                Node carNode = carsList.item(j);
                if (carNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element carElement = (Element) carNode;
                    if
(carElement.getAttribute("transaction").equals(transactionId)) {
                        double rentPrice =
Double.parseDouble(carElement.getElementsByTagName("rent_price").item(0).getTextConter
                        if (rentPrice > averageRent) {
                            outputBuilder.append(String.format("
<Transaction transaction_id=\"%s\">\n", transactionId));
                            outputBuilder.append(String.format("
<RentPrice>%s</RentPrice>\n", rentPrice));
                            outputBuilder.append("  </Transaction>\n");
                        }
                    }
                }
            }
        }
    }
    outputBuilder.append("</TransactionsAboveAverageRent>\n");

    // Lek rdez s azon aut kra , amelyeket 'Female' nem    gyfelek
b reltek ,  s  m g rendelkez sre  llnak
    outputBuilder.append("\n<AvailableCarsRentedByFemaleCustomers>\n");

    for (int i = 0; i < carsList.getLength(); i++) {
        Node carNode = carsList.item(i);
        if (carNode.getNodeType() == Node.ELEMENT_NODE) {
            Element carElement = (Element) carNode;
            String carStatus =
carElement.getElementsByTagName("car_status").item(0).getTextContent();
```

```
                String transactionId = carElement.getAttribute("transaction");

                if (carStatus.equals("Available")) {
                    for (int j = 0; j < customerList.getLength(); j++) {
                        Node customerNode = customerList.item(j);
                        if (customerNode.getNodeType() == Node.ELEMENT_NODE) {
                            Element customerElement = (Element) customerNode;
                            if
   (customerElement.getAttribute("transaction").equals(transactionId)) {
                                String gender =
   customerElement.getElementsByTagName("gender").item(0).getTextContent();
                                if (gender.equals("Female")) {
                                    String carModel =
   carElement.getElementsByTagName("car_modell").item(0).getTextContent();
                                    outputBuilder.append(String.format("  <Car
   model=\"%s\" transaction_id=\"%s\">\n", carModel, transactionId));
                                    outputBuilder.append(String.format("
   <Status>%s</Status>\n", carStatus));
                                    outputBuilder.append("  </Car>\n");
                                }
                            }
                        }
                    }
                }
            }
        }
        outputBuilder.append("</AvailableCarsRentedByFemaleCustomers>\n");

        System.out.println(outputBuilder);
    }
}
```

Programkód 3.3. DOMQueryIY5AM2.java adatlekérdező program

3.4. Adatírás

A kód fő funkciója, hogy beolvassa és kiírja az XML tartalmakat a konzolra és fájlba is. A `main` metódusban az *XMLIY5AM2* fájlt olvassa be a megadott útvonalról, használva a `DocumentBuilderFactory` és `DocumentBuilder` osztályokat az XML struktúra elemzéséhez. Ezután normalizálja a dokumentumot a `normalize` metódussal, ami segít a DOM fa struktúrájának rendezésében.

A kiíratást a `TransformerFactory` és a `Transformer` osztályok segítségével végezzük el, pont ahogy az adatmódosító kódban.

A `writeDocumentToFile` metódus is egy `Transformer` objektumot használ az XML dokumentum fájlba írásához, tiszteletben tartva az XML formázási szabályokat. Ez a metódus lehetővé teszi

egy új XML dokumentumok mentését. A kiírási folyamat során a kód biztosítja az XML szabványoknak megfelelő indentálást és formázást, ami olvashatóbbá és könnyebben értelmezhetővé teszi a kimenetet.

A kivételkezelés a kódban biztosítja, hogy az XML olvasás vagy írás közben fellépő hibák megfelelően kezelve legyenek, megelőzve ezzel a program összeomlását.

```
package domparse.IY5AM2;

import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import java.io.*;

public class DOMWriteIY5AM2 {

    public static void main(String[] args) {
        try {
            DocumentBuilderFactory dbFactory =
   DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.newDocument();

            // Gy k r elem l trehoz sa
            Element rootElement = doc.createElement("Car_IY5AM2");
            doc.appendChild(rootElement);

            // Namespace attrib tumok hozz ad sa
            rootElement.setAttribute("xmlns:xsi",
   "http://www.w3.org/2001/XMLSchema-instance");
            rootElement.setAttribute("xsi:noNamespaceSchemaLocation",
   "IY5AM2XSD.xsd");

            // Transaction elemek hozz ad sa
            addTransaction(doc, rootElement, "1", "Tes mnak", "2023-11-14",
   "Cash");
            addTransaction(doc, rootElement, "2", " zleti ", "2023-11-15",
   "Bank Transfer");
            addTransaction(doc, rootElement, "3", "Sz let snapi Aj nd k",
   "2023-11-16", "Credit Card");

            // Cars elemek hozz ad sa
            addCar(doc, rootElement, "1", "1", "1", "Opel", "200.0", "69",
   "Available");
            addCar(doc, rootElement, "3", "2", "3", "Toyota", "150.0", "33",
   "Unavailable");
```

```
        addCar(doc, rootElement, "4", "3", "2", "BMW", "300.0", "44",
"Available");

        // Driver elemek hozz ad sa
        addDriver(doc, rootElement, "1", "06501311010", "Male", "3534",
"Miskolc", "  rpd  ", "Buha", "Mil n");
        addDriver(doc, rootElement, "2", "06502223344", "Male", "1122",
"Debrecen", "Kossuth", "Szab ", "L szl ");
        addDriver(doc, rootElement, "3", "06505556677", "Female", "2045",
"T r kb lint", "F ", "Nagy", "Anna");

        // Customer elemek hozz ad sa
        addCustomer(doc, rootElement, "1", "1", "Female", "20",
"barbara@gmail.com", "3535", "Miskolc", "Kuruc", "Valaki", "Barbara");
        addCustomer(doc, rootElement, "2", "2", "Male", "30",
"istvan@gmail.com", "6000", "Kecskem t", "Pet fi", "Istv n", "Szab ");
        addCustomer(doc, rootElement, "3", "3", "Female", "25",
"eszter@gmail.com", "6722", "Szeged", "J zsef", "Eszter", "Kov cs");

        // Admin elemek hozz ad sa
        addAdmin(doc, rootElement, "1", "Male", "32", "admin@gmail.com",
"3634", "Valahol", "Napf ny", "Kiss", "J nos");
        addAdmin(doc, rootElement, "2", "Female", "28",
"admin2@gmail.com", "7632", "P cs", "Hunyadi", "T th", "Krisztina");
        addAdmin(doc, rootElement, "3", "Male", "40", "admin3@gmail.com",
"1012", "Budapest", "V r", "Varga", "Mih ly");

        // Approving elemek hozz ad sa
        addApproving(doc, rootElement, "1", "1", "1", "2023-11-14");
        addApproving(doc, rootElement, "2", "2", "2", "2023-11-15");
        addApproving(doc, rootElement, "3", "3", "3", "2023-11-16");

        // XML f jl   rsa
        TransformerFactory transformerFactory =
TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        DOMSource source = new DOMSource(doc);

        // XML f jl ment se
        StreamResult fileResult = new StreamResult(new
File("C:\\Users\\buha3\\Desktop\\XMLTaskIY5AM2\\IY5AM2XML_WRITE.xml"));
        transformer.transform(source, fileResult);

        // XML konzolra   rsa
        StreamResult consoleResult = new StreamResult(System.out);
        transformer.transform(source, consoleResult);
```

```
            System.out.println("The content has been written to the output
    file and the console successfully.");
        } catch (ParserConfigurationException | TransformerException e) {
            System.out.println("Error: " + e);
        }
    }

    // Seg df ggv ny az elemek hozz ad s hoz
    private static void addTransaction(Document doc, Element root, String id,
    String name, String date, String data) {
        Element transaction = doc.createElement("Transaction");
        transaction.setAttribute("transaction_id", id);
        root.appendChild(transaction);

        Element transactionName = doc.createElement("transaction_name");
        transactionName.appendChild(doc.createTextNode(name));
        transaction.appendChild(transactionName);

        Element transactionDate = doc.createElement("transaction_date");
        transactionDate.appendChild(doc.createTextNode(date));
        transaction.appendChild(transactionDate);

        Element transactions = doc.createElement("transactions");
        transaction.appendChild(transactions);

        Element transactionData = doc.createElement("transaction_data");
        transactionData.appendChild(doc.createTextNode(data));
        transactions.appendChild(transactionData);
    }

    // Seg df ggv ny az elemek hozz ad s hoz
    private static void addCar(Document doc, Element root, String carId,
    String transaction, String driver, String model, String price, String
    number, String status) {
        Element car = doc.createElement("Cars");
        car.setAttribute("car_id", carId);
        car.setAttribute("transaction", transaction);
        car.setAttribute("driver", driver);
        root.appendChild(car);

        Element carModel = doc.createElement("car_modell");
        carModel.appendChild(doc.createTextNode(model));
        car.appendChild(carModel);

        Element rentPrice = doc.createElement("rent_price");
        rentPrice.appendChild(doc.createTextNode(price));
        car.appendChild(rentPrice);
```

```
        Element carNumber = doc.createElement("car_number");
        carNumber.appendChild(doc.createTextNode(number));
        car.appendChild(carNumber);

        Element carStatus = doc.createElement("car_status");
        carStatus.appendChild(doc.createTextNode(status));
        car.appendChild(carStatus);
    }

    // Seg df ggv ny az elemek hozz ad s hoz
    private static void addDriver(Document doc, Element root, String driverId,
    String phoneNumber, String gender, String postCode, String city, String
    street, String firstName, String lastName) {
        Element driver = doc.createElement("Driver");
        driver.setAttribute("driver_id", driverId);
        root.appendChild(driver);

        Element phoneElement = doc.createElement("phone_number");
        phoneElement.appendChild(doc.createTextNode(phoneNumber));
        driver.appendChild(phoneElement);

        Element genderElement = doc.createElement("gender");
        genderElement.appendChild(doc.createTextNode(gender));
        driver.appendChild(genderElement);

        Element postCodeElement = doc.createElement("post_code");
        postCodeElement.appendChild(doc.createTextNode(postCode));
        driver.appendChild(postCodeElement);

        Element cityElement = doc.createElement("city");
        cityElement.appendChild(doc.createTextNode(city));
        driver.appendChild(cityElement);

        Element streetElement = doc.createElement("street");
        streetElement.appendChild(doc.createTextNode(street));
        driver.appendChild(streetElement);

        Element firstNameElement = doc.createElement("first_name");
        firstNameElement.appendChild(doc.createTextNode(firstName));
        driver.appendChild(firstNameElement);

        Element lastNameElement = doc.createElement("last_name");
        lastNameElement.appendChild(doc.createTextNode(lastName));
        driver.appendChild(lastNameElement);
    }

    // Seg df ggv ny az elemek hozz ad s hoz
    private static void addCustomer(Document doc, Element root, String
```

```
customerId, String transaction, String gender, String age, String email,
String postCode, String city, String street, String firstName, String
lastName) {
     Element customer = doc.createElement("Customer");
     customer.setAttribute("customer_id", customerId);
     customer.setAttribute("transaction", transaction);
     root.appendChild(customer);

     Element genderElement = doc.createElement("gender");
     genderElement.appendChild(doc.createTextNode(gender));
     customer.appendChild(genderElement);

     Element ageElement = doc.createElement("age");
     ageElement.appendChild(doc.createTextNode(age));
     customer.appendChild(ageElement);

     Element emailElement = doc.createElement("cust_email");
     emailElement.appendChild(doc.createTextNode(email));
     customer.appendChild(emailElement);

     Element postCodeElement = doc.createElement("post_code");
     postCodeElement.appendChild(doc.createTextNode(postCode));
     customer.appendChild(postCodeElement);

     Element cityElement = doc.createElement("city");
     cityElement.appendChild(doc.createTextNode(city));
     customer.appendChild(cityElement);

     Element streetElement = doc.createElement("street");
     streetElement.appendChild(doc.createTextNode(street));
     customer.appendChild(streetElement);

     Element firstNameElement = doc.createElement("first_name");
     firstNameElement.appendChild(doc.createTextNode(firstName));
     customer.appendChild(firstNameElement);

     Element lastNameElement = doc.createElement("last_name");
     lastNameElement.appendChild(doc.createTextNode(lastName));
     customer.appendChild(lastNameElement);
 }

 // Seg df ggv ny az elemek hozz ad s hoz
 private static void addAdmin(Document doc, Element root, String adminId,
String gender, String age, String email, String postCode, String city,
String street, String firstName, String lastName) {
     Element admin = doc.createElement("Admin");
     admin.setAttribute("admin_id", adminId);
     root.appendChild(admin);
```

```
        Element genderElement = doc.createElement("gender");
        genderElement.appendChild(doc.createTextNode(gender));
        admin.appendChild(genderElement);

        Element ageElement = doc.createElement("age");
        ageElement.appendChild(doc.createTextNode(age));
        admin.appendChild(ageElement);

        Element emailElement = doc.createElement("admin_email");
        emailElement.appendChild(doc.createTextNode(email));
        admin.appendChild(emailElement);

        Element postCodeElement = doc.createElement("post_code");
        postCodeElement.appendChild(doc.createTextNode(postCode));
        admin.appendChild(postCodeElement);

        Element cityElement = doc.createElement("city");
        cityElement.appendChild(doc.createTextNode(city));
        admin.appendChild(cityElement);

        Element streetElement = doc.createElement("street");
        streetElement.appendChild(doc.createTextNode(street));
        admin.appendChild(streetElement);

        Element firstNameElement = doc.createElement("first_name");
        firstNameElement.appendChild(doc.createTextNode(firstName));
        admin.appendChild(firstNameElement);

        Element lastNameElement = doc.createElement("last_name");
        lastNameElement.appendChild(doc.createTextNode(lastName));
        admin.appendChild(lastNameElement);
    }

    // Seg df ggv ny az elemek hozz ad s hoz
    private static void addApproving(Document doc, Element root, String
   approvingId, String adminId, String transactionId, String date) {
        Element approving = doc.createElement("Approving");
        approving.setAttribute("approving_id", approvingId);
        approving.setAttribute("admin", adminId);
        approving.setAttribute("transaction", transactionId);
        root.appendChild(approving);

        Element dateElement = doc.createElement("verification_date");
        dateElement.appendChild(doc.createTextNode(date));
        approving.appendChild(dateElement);
    }
```

```
}
```

Programkód 3.4. DOMWriteIY5AM2.java adatíró program