JEGYZŐKÖNYV

Webes adatkezelő környezetek

Féléves feladat

Restaurant

Készítette: **Sziráczki Soma**
Neptunkód: **BK6QE8**

Dátum: 2024.12.07

Tartalom

1. A feladat témája

Az étterem működését segítő olyan komplex nyilvántartási rendszer kidolgozása és megvalósítása, amely részletesen tartalmazza az étterem által kiszállított rendelések adatait. A rendszer célja, hogy átláthatóvá tegye a rendelések kezelését, nyomon követhetővé tegye a kiszállításokat, és támogassa az alapanyagok pontos adminisztrációját. A nyilvántartásban szerepelnie kell a rendelésekhez kapcsolódó részletes információknak, például az egyes megrendelések tételeinek, a rendelés dátumának, az ügyfél adatainak, valamint a szállítás státuszának. Ezen kívül fontos, hogy a rendszer tartalmazza a futárok részletes listáját, beleértve a nevüket, elérhetőségeiket, valamint azt, hogy mely rendelések szállításáért feleltek.

A nyilvántartásnak figyelembe kell vennie az egyes ételek elkészítéséhez szükséges alapanyagokat is, ezek pontos nyilvántartását, valamint az alapanyagok rendelkezésre állását és felhasználását. Ezáltal a rendszer segíthet az alapanyag-utánpótlás megtervezésében és az étterem hatékony működésében. Összességében a nyilvántartás célja, hogy teljes körű adatkezelést biztosítson az étterem logisztikai és adminisztrációs feladataihoz, miközben támogatja a gördülékeny és pontos szolgáltatásnyújtást.

Ingredients egyed tulajdonságai

- ingredient_id: az alapanyag egyedi azonosítója
- name: az alapanyag neve
- type: az alapanyag típusa pl.: feltét
- purchasePrice: mennyiért lehet beszerezni az alapanyagot
- stockQuantity: mennyi van az alapanyagból raktáron

Product egyed tulajdonságai

- product_id: az egyed egyedi azonosítója
- name: a termék neve
- price: a termék ára
- type: többértékű tulajdonság ami a termék típusát adja meg
- description: a termék leírása

Order egyed tulajdonságai

- order_id: a megrendelés egyedi kulcsa
- date: a megrendelés ideje
- price: a megrendelés végösszege
- items: a megrendelt termékek, többértékű tulajdonság
- status: a megrendelés állapota

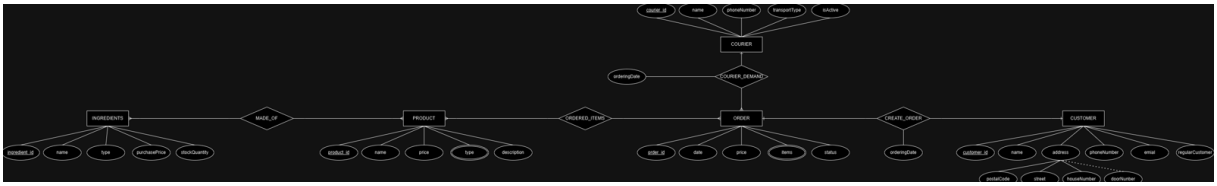
Courier egyed tulajdonságai

- courier_id: a futár egyedi azonosítója
- name: a futár neve
- phoneNumber: a futár telefonszáma
- transportType: milyen jármű segítségével szállít a futár
- isActive: dolgozik e éppen a futár

Customer egyed tulajdonságai

- customer_id: a vevő kulcsa
- name: a vevő neve
- address: összetett tulajdonság, a vevő címe
- phoneNumber: a vevő telefonszáma
- email: a vevő email címe
- regularCustomer: törzsvendég-e

A feladat ER modellje:



Az egyedek között lévő kapcsolatok:

Product és Order közötti kapcsolat: Ordered_Items

- Több-több kapcsolat van közöttük, mivel egy termék többször is szerepelhet egy rendelésben, illetve egy rendeléshez több termék is tartozhat

Ingredients és Product közötti kapcsolat: Made_Of

- Több-több kapcsolat van közöttük, mivel egy alapanyagot több termékhez is fel lehet használni, illetve több alapanyagot is lehet egy termékhez.

Order és Courier közötti kapcsolat: Courier_Demand

- Egy-több kapcsolat van közöttük, mivel több rendelést is tud egy futár kiszállítani.

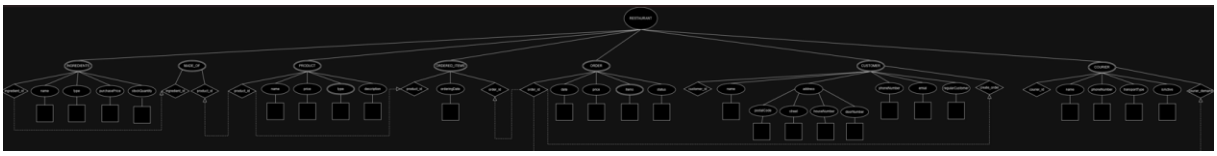
Order és Customer közötti kapcsolat: Create_Order

- Egy-egy kapcsolat van közöttük, mivel egy vevőhöz egy rendelés tartozhat csak, ami aktív, illetve egy megrendelés csak egy vevőhöz tartozik.

2. Az ER modell konvertálása XDM modellé

Az XDM modell használata során háromféle szimbólumot alkalmazunk: ellipszist, rombuszt és téglalapot. Az ellipszisek az elemeket reprezentálják, amelyek minden egyes egyedből származnak, és magukba foglalják a hozzájuk tartozó tulajdonságokat is. A rombuszok az attribútumokat jelölik, amelyek a kulcstulajdonságokból származnak. A téglalapok pedig a szöveges tartalmat képviselik, amely az XML dokumentumban jelenik meg.
Azokat az elemeket, amelyek többször is előfordulhatnak, dupla ellipszissel ábrázoljuk. A kulcsok és idegen kulcsok közötti kapcsolatokat szaggatott vonallal összekötött nyíl mutatja.

A feladat XDM modellje:



3. XML dokumentum készítése

Az XDM modell alapján a következő lépésben elkészítettem az XML dokumentumot. Elsőként a gyökérelemet, a Restaurant elemet hoztam létre. Ezután a gyökérelem többértékű gyermekelemeiből legalább három-három példányt készítettem. Minden elemhez hozzárendeltem az attribútumokat, amelyek tartalmazták a kulcsokat és idegen kulcsokat. Ezekhez az attribútumokhoz további gyermekelemeket is készítettem, amelyek között szerepeltek többértékű és összetett értékű elemek is.

XML dokumentum forráskódja:

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?><Restaurant_BK6QE8
xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="XMLSchema_BK6QE8.xsd">

    <!--Ingredients-->

    <ingredients ingredient_id="1">
        <name>Flour</name>
        <type>Base</type>
        <purchasePrice>500</purchasePrice>
        <stockQuantity>1000</stockQuantity>
    </ingredients>

    <ingredients ingredient_id="2">
        <name>Honey</name>
```

```
    <type>Sweetener</type>
    <purchasePrice>800</purchasePrice>
    <stockQuantity>500</stockQuantity>
</ingredients>

<ingredients ingredient_id="3">
    <name>Cheese</name>
    <type>Dairy</type>
    <purchasePrice>400</purchasePrice>
    <stockQuantity>300</stockQuantity>
</ingredients>

<ingredients ingredient_id="4">
    <name>Tomatoes</name>
    <type>Vegetable</type>
    <purchasePrice>200</purchasePrice>
    <stockQuantity>400</stockQuantity>
</ingredients>

<ingredients ingredient_id="5">
    <name>Beef</name>
    <type>Meat</type>
    <purchasePrice>1500</purchasePrice>
    <stockQuantity>200</stockQuantity>
</ingredients>

<ingredients ingredient_id="6">
    <name>Salmon</name>
    <type>Fish</type>
```

```
        <purchasePrice>1800</purchasePrice>
        <stockQuantity>150</stockQuantity>
    </ingredients>

    <!--Products-->

    <product product_id="1">
        <name>Spaghetti Bolognese</name>
        <price>2400</price>
        <type>Pasta</type>
        <description>Classic spaghetti with Bolognese sauce.</description>
    </product>

    <product product_id="2">
        <name>Greek Salad</name>
        <price>1800</price>
        <type>Salad</type>
        <description>A refreshing mix of cucumbers, tomatoes, olives, and feta cheese.</description>
    </product>

    <product product_id="3">
        <name>Grilled Salmon</name>
        <price>4000</price>
        <type>American</type>
        <description>Perfectly grilled salmon fillet with a side of sautéed tomatoes.</description>
    </product>
```

<!--Made of kapcsolat-->

<made_of ingredient_id="1" product_id="1"/>
<made_of ingredient_id="4" product_id="1"/>
<made_of ingredient_id="5" product_id="1"/>

<made_of ingredient_id="4" product_id="2"/>
<made_of ingredient_id="3" product_id="2"/>

<made_of ingredient_id="6" product_id="3"/>
<made_of ingredient_id="4" product_id="3"/>

<!--Orders-->

<order order_id="1">
    <date>2024.04.10</date>
    <price>4800</price>
    <item>Spaghetti Bolognese</item>
    <item>Greek Salad</item>
    <status>Completed</status>
</order>

<order order_id="2">
    <date>2024.05.20</date>
    <price>1800</price>
    <item>Greek Salad</item>
    <status>Processing</status>
</order>

```
<order order_id="3">
    <date>2024.06.15</date>
    <price>5800</price>
    <item>Grilled Salmon</item>
    <status>Shipping</status>
</order>

<!--Order items kapcsolat-->

<ordered_items order_id="1" product_id="1">
    <orderingDate>2024.04.10</orderingDate>
</ordered_items>

<ordered_items order_id="2" product_id="2">
    <orderingDate>2024.05.20</orderingDate>
</ordered_items>

<ordered_items order_id="3" product_id="3">
    <orderingDate>2024.06.15</orderingDate>
</ordered_items>

<!--Customers-->

<customer create_order="1" customer_id="1">
    <name>Emma Brown</name>
    <address>
        <postalcode>90210</postalcode>
        <street>Elm Street</street>
        <houseNumber>12</houseNumber>
```

```
        <doorNumber>4</doorNumber>
    </address>
    <phoneNumber>+ 16505556789</phoneNumber>
    <email>emma.brown@example.com</email>
    <regularCustomer>true</regularCustomer>
</customer>

<customer create_order="2" customer_id="2">
    <name>Liam Davis</name>
    <address>
        <postalcode>10001</postalcode>
        <street>Broadway Avenue</street>
        <houseNumber>7</houseNumber>
        <doorNumber>1</doorNumber>
    </address>
    <phoneNumber>+ 14155552345</phoneNumber>
    <email>liam.davis@example.com</email>
    <regularCustomer>false</regularCustomer>
</customer>

<customer create_order="3" customer_id="3">
    <name>Olivia Taylor</name>
    <address>
        <postalcode>30303</postalcode>
        <street>Peachtree Street</street>
        <houseNumber>5</houseNumber>
        <doorNumber>3</doorNumber>
    </address>
    <phoneNumber>+ 17705556789</phoneNumber>
```

```
        <email>olivia.taylor@example.com</email>
        <regularCustomer>true</regularCustomer>
    </customer>

    <!--Couriers-->

    <courier courier_demand="1" courier_id="1">
        <name>James Wilson</name>
        <phoneNumber>+ 15555554321</phoneNumber>
        <transportType>Bicycle</transportType>
        <isActive>true</isActive>
    </courier>

    <courier courier_demand="2" courier_id="2">
        <name>Charlotte White</name>
        <phoneNumber>+ 14445555678</phoneNumber>
        <transportType>Car</transportType>
        <isActive>true</isActive>
    </courier>

    <courier courier_demand="3" courier_id="3">
        <name>Henry Moore</name>
        <phoneNumber>+ 13335556666</phoneNumber>
        <transportType>Motorcycle</transportType>
        <isActive>true</isActive>
    </courier>

</Restaurant_BK6QE8>
```

4. XMLSchema készítése

Az XML dokumentum validálásához létre kellett hoznom egy XMLSchema sémát. Első lépésként definiáltam az egyszerű elemeket, amelyekre később hivatkozni tudok, valamint ezekhez saját adattípusokat is rendeltem. Az adattípusok meghatározásánál használtam enumerációkat és reguláris kifejezéseket is. Ezután elkészítettem a séma tényleges szerkezetét, amelyben hivatkoztam az előzőleg definiált egyszerű elemekre.
A séma felépítése során beállítottam az elemek minimum és maximum előfordulási számát is, különös tekintettel a többértékű elemekre, ahol ez elengedhetetlen. Végül definiáltam az attribútumokat, valamint a kulcsokat, idegen kulcsokat, és az 1:1 kapcsolati struktúrákat is, hogy a séma teljes mértékben megfeleljen az XML dokumentum követelményeinek.

Az XMLSchema kódja:

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

    <!--Egyszerű elemek-->

    <xs:element name="name" type="xs:string" />
    <xs:element name="type" type="ingredients_food_type" />
    <xs:element name="purchasePrice" type="xs:int" />
    <xs:element name="stockQuantity" type="xs:int" />

    <xs:element name="price" type="xs:int" />
    <xs:element name="description" type="xs:string" />

    <xs:element name="date" type="idoTipus" />
    <xs:element name="item" type="xs:string" />
    <xs:element name="status" type="status_type" />
```

```
<xs:element name="orderingDate" type="idoTipus" />

<xs:element name="postalcode" type="xs:int" />
<xs:element name="street" type="xs:string" />
<xs:element name="houseNumber" type="xs:int" />
<xs:element name="doorNumber" type="xs:int" />
<xs:element name="email" type="emailTipus" />
<xs:element name="phoneNumber" type="telefonszamTipus" />

<xs:element name="transportType" type="xs:string" />
<xs:element name="isActive" type="xs:string" />

<!--Saját típusok-->

<xs:simpleType name="ingredients_food_type">
   <xs:restriction base="xs:string">
      <xs:enumeration value="Base" />
      <xs:enumeration value="Sweetener" />
      <xs:enumeration value="Dairy" />
      <xs:enumeration value="Vegetable" />
      <xs:enumeration value="Meat" />
      <xs:enumeration value="Fish" />
      <xs:enumeration value="Pasta" />
      <xs:enumeration value="Italian" />
      <xs:enumeration value="Salad" />
      <xs:enumeration value="American" />
      <xs:enumeration value="Japanese" />
      <xs:enumeration value="Sushi" />
```

```
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="status_type">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Shipping" />
      <xs:enumeration value="Processing" />
      <xs:enumeration value="Completed" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="telefonszamTipus">
    <xs:restriction base="xs:string">
      <xs:pattern value="(06(20|30|31|50|60|70)₩d{7})"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="idoTipus">
    <xs:restriction base="xs:string">
      <xs:pattern value="([12]₩d{3}.(0[1-9]|1[0-2]).(0[1-
9]|[12]₩d|3[01]))" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="emailTipus">
    <xs:restriction base="xs:string">
      <xs:pattern value="[₩w₩.]+ @([₩w]+ ₩.)+ [₩w]{2,4}" />
    </xs:restriction>
  </xs:simpleType>
```

```
    <!--Felépítés-->

    <xs:element name="Restaurant_BK6QE8">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="ingredients" minOccurs="1"
maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="name" />
                            <xs:element name="type" />
                            <xs:element ref="purchasePrice" />
                            <xs:element ref="stockQuantity" />
                        </xs:sequence>
                        <xs:attribute name="ingredient_id" type="xs:int" />
                    </xs:complexType>
                </xs:element>
                <xs:element name="product" minOccurs="1"
maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="name" />
                            <xs:element ref="price" />
                            <xs:element ref="type" minOccurs="1"
maxOccurs="unbounded"/>
                            <xs:element ref="description" />
                        </xs:sequence>
                        <xs:attribute name="product_id" type="xs:int" />
```

```
            </xs:complexType>
        </xs:element>
        <xs:element name="made_of" minOccurs="1"
maxOccurs="unbounded">
            <xs:complexType>
                <xs:attribute name="ingredient_id" type="xs:int" />
                <xs:attribute name="product_id" type="xs:int" />
            </xs:complexType>
        </xs:element>
        <xs:element name="order" minOccurs="1"
maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="date" />
                    <xs:element ref="price" />
                    <xs:element ref="item" minOccurs="1"
maxOccurs="unbounded" />
                    <xs:element ref="status" />
                </xs:sequence>
                <xs:attribute name="order_id" type="xs:int" />
            </xs:complexType>
        </xs:element>
        <xs:element name="ordered_items" minOccurs="1"
maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="orderingDate" />
                </xs:sequence>
                <xs:attribute name="product_id" type="xs:int" />
```

```
                <xs:attribute name="order_id" type="xs:int" />
            </xs:complexType>
        </xs:element>
        <xs:element name="customer" minOccurs="1"
maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="name" />
                    <xs:element name="address" minOccurs="1"
maxOccurs="1" >
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="postalcode" />
                                <xs:element ref="street" />
                                <xs:element ref="houseNumber" />
                                <xs:element ref="doorNumber" />
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="phoneNumber" />
                    <xs:element name="email" />
                    <xs:element name="regularCustomer" />
                </xs:sequence>
                <xs:attribute name="customer_id" type="xs:int" />
                <xs:attribute name="create_order" type="xs:int" />
            </xs:complexType>
        </xs:element>
        <xs:element name="courier" minOccurs="1"
maxOccurs="unbounded">
```

```
            <xs:complexType>
               <xs:sequence>
                  <xs:element ref="name" />
                  <xs:element name="phoneNumber" />
                  <xs:element name="transportType" />
                  <xs:element name="isActive" />
               </xs:sequence>
               <xs:attribute name="courier_id" type="xs:int" />
               <xs:attribute name="courier_demand" type="xs:int" />
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>

   <!--Kulcsok-->

   <xs:key name="ingredient_kulcs">
      <xs:selector xpath="ingredients" />
      <xs:field xpath="@ingredient_id" />
   </xs:key>

   <xs:key name="product_kulcs">
      <xs:selector xpath="product" />
      <xs:field xpath="@product_id" />
   </xs:key>

   <xs:key name="order_kulcs">
      <xs:selector xpath="order" />
      <xs:field xpath="@order_id" />
```

```
        </xs:key>

        <xs:key name="customer_kulcs">
            <xs:selector xpath="customer" />
            <xs:field xpath="@customer_id" />
        </xs:key>

        <xs:key name="courier_kulcs">
            <xs:selector xpath="courier" />
            <xs:field xpath="@courier_id" />
        </xs:key>

        <!--Idegen kulcsok-->

        <xs:keyref refer="ingredient_kulcs"
name="made_of_ingredient_idegen_kulcs">
            <xs:selector xpath="made_of" />
            <xs:field xpath="@ingredient_id" />
        </xs:keyref>

        <xs:keyref refer="product_kulcs"
name="made_of_product_idegen_kulcs">
            <xs:selector xpath="made_of" />
            <xs:field xpath="@product_id" />
        </xs:keyref>

        <xs:keyref refer="order_kulcs" name="ordered_items_idegen_kulcs">
            <xs:selector xpath="ordered_items" />
            <xs:field xpath="@order_id" />
```

```
        </xs:keyref>

        <xs:keyref refer="order_kulcs" name="create_order_idegen_kulcs">
            <xs:selector xpath="customer" />
            <xs:field xpath="@create_order" />
        </xs:keyref>

        <xs:keyref refer="order_kulcs" name="courier_demand_idegen_kulcs">
            <xs:selector xpath="courier" />
            <xs:field xpath="@courier_demand" />
        </xs:keyref>

        <!--1:1-->

        <xs:unique name="create_order">
            <xs:selector xpath="customer" />
            <xs:field xpath="@create_order" />
        </xs:unique>

    </xs:element>

</xs:schema>
```

5. DOM adatolvasás

A feladat megkezdésekor először meg kellett határoznom, melyik fájlból történjen a beolvasás. Ezt a fájlt a projekt mappájában helyeztem el. A beolvasáshoz létrehoztam egy

dokumentum objektumot, amely lehetővé tette a gyökérelem lekérdezését, valamint ezt követően a szülő- és gyermekelemek elérését.
A többször előforduló elemek kezeléséhez NodeList-et hoztam létre, amelyben ezek az elemek tárolhatók. Miután ez elkészült, végigiteráltam az elemek listáján, hogy kinyerhessem belőlük a különböző attribútumokat és gyermekelemeket, amelyeket később kiírtam. Az egyes gyermekelemek között lehettek olyanok is, amelyek többértékűek voltak, ezért megvizsgáltam, hogy ezekből az elemekből több van-e, vagy csak egy. Ha több elem volt, akkor további for ciklussal iteráltam rajtuk, és szintén kiírtam az értéküket.

Kód:

```
package hu.domparse.bk6qe8;



import java.io.File;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class DOMReadBK6QE8 {

    public static void main(String[] args) {
        try {

            String xmlFilePath = "C:₩₩Users₩₩Sziráczki
Soma₩₩Desktop₩₩XML₩₩XMLTask_BK6QE8₩₩1.
Feladat₩₩XML_BK6QE8.xml";

            // XML dokumentum létrehozása és beolvasása
```

```
            File xmlFile = new File(xmlFilePath);
            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(xmlFile);

            // Dokumentum elemeinek kiírása a konzolra
            printNode(doc.getDocumentElement(), "");

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

    // Rekurzív metódus az XML elemek kiírására
    private static void printNode(Node node, String behúzás) {

        if (node.getNodeType() == Node.ELEMENT_NODE) {

            //elemek elott enter
            if (node.getParentNode().getNodeType() == Node.ELEMENT_NODE)
{

                System.out.println("");

            }

            //tagek kiírása
            System.out.println(behúzás + "<" + node.getNodeName() + ">");
```

```
            //tagnév kiírása
            NodeList nodeList = node.getChildNodes();
            for (int i = 0; i < nodeList.getLength(); i++) {
                printNode(nodeList.item(i), behúzás + "  ");
            }

            //tagek lezárása
            System.out.println(behúzás + "</" + node.getNodeName() + ">");

            //tag után üres sor
            System.out.println("");

        } else if (node.getNodeType() == Node.TEXT_NODE) {

            //tag szöveg tartalmának kiírása
            String text = node.getNodeValue().trim();
            if (!text.isEmpty()) {
                System.out.println(behúzás + text);
            }

        } else if (node.getNodeType() == Node.COMMENT_NODE) {

            //comment kiírása
            System.out.println("");
            System.out.println(behúzás + "<!-- " + node.getNodeValue() + " --
>");

        }
```

```
    }
}
```

6. DOM adatmódosítás

Az első ordernek a kulcsát választottam ki módosításra. Ez a kulcs attribútum több helyen is megjelenik mint például a customer, courier elemeknél is, ezért ezeket ott is megváltoztattam, illetve a különböző kapcsolati elemeknél is. A módosított fájlt kiírattam a konzolra, illetve egy új fájlba mentem.
Kód:

```
package hu.domparse.bk6qe8;

import java.io.File;
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
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.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
```

```
public class DOMModifyBK6QE8 {

	public static void main(String argv[]) throws SAXException,
IOException, ParserConfigurationException {

		try {

			File inputFile = new File("C:₩₩Users₩₩Szirácz ki
Soma₩₩Desktop₩₩XML₩₩XMLTask_BK6QE8₩₩1.
Feladat₩₩XML_BK6QE8_toModify.xml");

			DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
			DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();

			Document doc = documentBuilder.parse(inputFile);

			// order attribútumának módosítása
			Node csoport =
doc.getElementsByTagName("order").item(0);

			NamedNodeMap attr = csoport.getAttributes();
			Node nodeAttr = attr.getNamedItem("order_id");
			nodeAttr.setTextContent("4");

			// order item módosítás
			NodeList kszList =
doc.getElementsByTagName("ordered_items");
```

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

			if (node.getNodeType() == Node.ELEMENT_NODE) {

				Element eElement = (Element) node;
				String tagValue =
eElement.getAttribute("order_id");

				if ("1".equals(tagValue)) {
					eElement.setAttribute("order_id", "4");
				}

			}
		}

		// customer items módosítás
		NodeList cList = doc.getElementsByTagName("customer");

		for (int i = 0; i < cList.getLength(); i++) {
			Node node = cList.item(i);

			if (node.getNodeType() == Node.ELEMENT_NODE) {

				Element eElement = (Element) node;
				String tagValue =
eElement.getAttribute("create_order");
```

```
                        if ("1".equals(tagValue)) {
                                eElement.setAttribute("create_order",
"4");
                        }

                }
        }

        // courier item módosítás
        NodeList courList =
doc.getElementsByTagName("courier");

        for (int i = 0; i < courList.getLength(); i++ ) {
                Node node = courList.item(i);

                if (node.getNodeType() == Node.ELEMENT_NODE) {

                        Element eElement = (Element) node;
                        String tagValue =
eElement.getAttribute("courier_demand");

                        if ("1".equals(tagValue)) {
                                eElement.setAttribute("courier_demand",
"4");
                        }

                }
        }
```

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

                DOMSource source = new DOMSource(doc);

                System.out.println("-Módosított fájl-");

                StreamResult consoleResult = new
StreamResult(System.out);
                StreamResult file = new StreamResult(inputFile);

                transformer.transform(source, consoleResult);
                transformer.transform(source, file);

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

}
```

7. DOM adatlekérdezés

Az XPath segítségével végeztem el a lekérdezéseket, összesen hét különböző lekérdezést készítettem. Ezeket alaposan leteszteltem, és a feltételeknek megfelelő eredményeket kiírtam a konzolra. A lekérdezések a következőket tartalmazták:

- A Restaurant_BK6QE8 gyökérelemhez tartozó alapanyag gyermekelemek lekérdezése.
- Azoknak az alapanyagoknak a kiválasztása, amelyek rendelkeznek attribútumokkal.
- Azok az alapanyagok, amelyek ára pontosan 500 Ft.
- Az alapanyagok típusának lekérdezése, amelyek ára meghaladja a 200 Ft-ot.
- A termékek közül az első két elem lekérdezése.
- Az a termék, amelynek azonosítója 2-es.
- A harmadik termék kiválasztása.

A kód:

```
package hu.domparse.bk6qe8;

import java.io.File;
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

public class DOMQueryBK6QE8 {
```

```
    public static void main(String[] args) {

        try {

            // DocumentBuilder
            DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder documentBuilder =
documentBuilderFactory.newDocumentBuilder();

            File xmlFile = new File("C:₩₩Users₩₩Szirá czki
Soma₩₩Desktop₩₩XML₩₩XMLTask_BK6QE8₩₩1.
Feladat₩₩XML_BK6QE8.xml");
            if (!xmlFile.exists()) {
                System.out.println("A fájl nem található: " +
xmlFile.getAbsolutePath());
                return;
            }
            Document document = documentBuilder.parse(xmlFile);



            document.getDocumentElement().normalize();

            // XPath
            XPath xPath = XPathFactory.newInstance().newXPath();

            // a Restaurant_BK6QE8 root elem alapanyag gyerekelemei
            String expression = "Restaurant_BK6QE8 / ingredients";
```

```
                // alapanyagok, amiknek van attribútuma
                // String expression = "//ingredients[@*]";

                // alapanyagok, amik mint 500ba kerülnek
                // String expression = "//ingredients[purchasePrice='500']";

                // alapanyagok típusa, amik több, mint 200Ft-ba kerülnek
                // String expression =
"//ingredients[purchasePrice>200]/type";

                // termékek elso két eleme
                // String expression = "//product[position()<3]";

                // 2-es azonositójú termék
                // String expression = "//product[@product_id='2']";

                // a harmadik termék kiválasztáas
                // String expression = "Restaurant_BK6QE8/product[3]";

                // keszitunk egy listat, majd az xPath kifejezest le kell
forditani es ki kell ertekelni
                NodeList nodeList = (NodeList)
xPath.compile(expression).evaluate(document, XPathConstants.NODESET);

                for (int i = 0; i < nodeList.getLength(); i++) {
                        Node node = nodeList.item(i);
                        System.out.println("₩nAktualis elem: " +
node.getNodeName());
```

```
                    if (node.getNodeType() == Node.ELEMENT_NODE
&& node.getNodeName().equals("ingredients")) {

                        Element elem = (Element) node;

                String id = elem.getAttribute("ingredient_id");

                Node node1 = elem.getElementsByTagName("name").item(0);
                String name = node1.getTextContent();

                Node node2 = elem.getElementsByTagName("type").item(0);
                String type = node2.getTextContent();

                Node node3 =
elem.getElementsByTagName("purchasePrice").item(0);
                String purchaseP = node3.getTextContent();

                        Node node4 =
elem.getElementsByTagName("stockQuantity").item(0);
                String stockQ = node4.getTextContent();

                System.out.println("Alapanyag id-je: " + id);
                System.out.println("Név: " + name);
                System.out.println("Típus: " + type);
                System.out.println("PurchasePrice: " + purchaseP);
                        System.out.println("StockQuantity: " +
stockQ);

                    }
```

```
                        // alapanyagok típusa
                        if (node.getNodeType() == Node.ELEMENT_NODE
&& node.getNodeName().equals("type")) {

                                Element element = (Element) node;

                                System.out.println("Alapanyag típusa: " +
element.getTextContent());

                        }

                        //product kiiratasa
                        if (node.getNodeType() == Node.ELEMENT_NODE
&& node.getNodeName().equals("product")) {

                                Element element = (Element) node;

                                System.out.println("ID: " +
element.getAttribute("product_id"));

                                System.out.println(
                                                "Termék neve: " +
element.getElementsByTagName("name").item(0).getTextContent());

                                System.out.println(
                                                "Ára: " +
element.getElementsByTagName("price").item(0).getTextContent());
```

```
                                        if
(nodeList.item(i).getChildNodes().getLength() > 3) {
                                                int db = 0;
                                                Node node4 =
element.getElementsByTagName("type").item(0);
                                                while (node4 != null) {
                                                        node4 =
element.getElementsByTagName("type").item(db);
                                                        if (node4 != null) {
                                                                String type =
node4.getTextContent();

        System.out.println("A típusa: " +  type);
                                                        }
                                                        db+ + ;
                                                }

                                        }

                                System.out.println(
                                        "Leírása: " +
element.getElementsByTagName("description").item(0).getTextContent());




                        }
                }
```

```
            } catch (ParserConfigurationException e) {
                e.printStackTrace();
            } catch (SAXException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (XPathExpressionException e) {
                e.printStackTrace();
            }

        }

}
```

8. DOM adatírás

Az XML-ben található minden gyermekelemhez létrehoztam egy metódust, amely lehetővé tette, hogy ezeket a gyökérelemhez hozzáadjam. A metódus paraméterein keresztül adtam meg a tagek nevét és a hozzájuk tartozó értékeket. Miután ezeket hozzáadtam, az eredményt mind a konzolra, mind pedig egy külön fájlba kiírtam. A helyes végrehajtást az XMLSchema segítségével ellenőriztem, amely biztosította, hogy az XML megfeleljen az előírt sémának.

A kód:

```
package hu.domparse.bk6qe8;

import java.io.File;

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

```
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.Comment;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;

public class DOMWriteBK6QE8 {

	public static void main(String argv[]) throws Exception {

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

		Document doc = dBuilder.newDocument();

		Element root = doc.createElementNS("BK6QE8",
"Restaurant_BK6QE8");
		doc.appendChild(root);

		// ingredients

		root.appendChild(createIngredients(doc, "1", "Flour", "Base",
"500", "1000"));
```

```
            root.appendChild(createIngredients(doc, "2", "Cheese", "Dairy",
"400", "300"));
            root.appendChild(createIngredients(doc, "3", "Salmon", "Fish",
"1800", "150"));

            Element element = (Element)
doc.getElementsByTagName("ingredients").item(0);
            Comment comment = doc.createComment("Ingredients");
            element.getParentNode().insertBefore(comment, element);

            // products

            String[] t = {"pasta", "italian"};
        root.appendChild(createProduct(doc, "1", "Spaghetti", "2400", t, "Classic
spaghetti with Bolognese sauce."));
        String[] t2 = {"salad"};
        root.appendChild(createProduct(doc, "2", "Greek Salad", "1800", t2, "A
refreshing mix of cucumbers, tomatoes, olives, and feta cheese."));
        String[] t3 = {"japanese", "sushi"};
        root.appendChild(createProduct(doc, "3", "Grilled Salmon", "4000", t3,
"Perfectly grilled salmon fillet with a side of sautéed tomatoes."));

            element = (Element)
doc.getElementsByTagName("product").item(0);
            comment = doc.createComment("Products");
            element.getParentNode().insertBefore(comment, element);

            // made_of
```

```
                root.appendChild(createMadeOf(doc, "1", "1"));
                root.appendChild(createMadeOf(doc, "2", "2"));
                root.appendChild(createMadeOf(doc, "3", "3"));

                element = (Element)
doc.getElementsByTagName("made_of").item(0);
                comment = doc.createComment("Made of kapcsolat");
                element.getParentNode().insertBefore(comment, element);

                // orders

                String[] i = {"Spaghetti Bolognese", "Greek Salad"};
                root.appendChild(createOrder(doc, "1", "2024.04.10", "4800",
i,"Completed"));

                String[] i2 = {"Sushi box"};
                root.appendChild(createOrder(doc, "2", "2024.05.20", "1800",
i2,"Processing"));

                String[] i3 = {"Spaghetti Bolognese"};
                root.appendChild(createOrder(doc, "3", "2024.06.15", "5800",
i3,"Shipping"));

                element = (Element)
doc.getElementsByTagName("order").item(0);
                comment = doc.createComment("Orders");
                element.getParentNode().insertBefore(comment, element);

                // ordered_items
```

```
            root.appendChild(createOrderedItems(doc, "1", "1",
"2024.04.10"));
            root.appendChild(createOrderedItems(doc, "2", "2",
"2024.05.20"));
            root.appendChild(createOrderedItems(doc, "3", "3",
"2024.06.15"));

            element = (Element)
doc.getElementsByTagName("ordered_items").item(0);
            comment = doc.createComment("Order items kapcsolat");
            element.getParentNode().insertBefore(comment, element);

            // customer

        root.appendChild(createCustomer(doc, "1", "1", "Emma Brown", "90210",
"Elm Street", "12", "4", "+16505556789", "emma.brown@example.com",
"true"));
        root.appendChild(createCustomer(doc, "2", "2", "Liam Davis", "10001",
"Broadway Avenue", "7", "1", "+14155552345", "liam.davis@example.com",
"false"));
        root.appendChild(createCustomer(doc, "3", "3", "Olivia Taylor", "30303",
"Peachtree Street", "5", "3", "+17705556789", "olivia.taylor@example.com",
"true"));

            element = (Element)
doc.getElementsByTagName("customer").item(0);
            comment = doc.createComment("Customers");
            element.getParentNode().insertBefore(comment, element);
```

```
            // courier

            root.appendChild(createCourier(doc, "1", "1", "James Wilson",
"+ 15555554321", "Bicycle", "true" ));
            root.appendChild(createCourier(doc, "2", "2", "Charlotte White",
"+ 14445555678", "Car", "true" ));
            root.appendChild(createCourier(doc, "3", "3", "Henry Moore",
"+ 13335556666", "Motorcycle", "true" ));

            element = (Element)
doc.getElementsByTagName("courier").item(0);
            comment = doc.createComment("Couriers");
            element.getParentNode().insertBefore(comment, element);

            // Transform

            TransformerFactory transformerFactory =
TransformerFactory.newInstance();
            Transformer transf = transformerFactory.newTransformer();

            transf.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
            transf.setOutputProperty(OutputKeys.INDENT, "yes");
            transf.setOutputProperty("{https://xml.apache.org/xslt}indent-
amount", "2");

            // File letrehozas

            DOMSource source = new DOMSource(doc);
```

```
            File myFile = new File("C:₩₩Users₩₩Szirácz
ki Soma₩₩Desktop₩₩XML₩₩XMLTask_BK6QE8₩₩2.
Feladat₩₩DOMParse_BK6QE8₩₩XML_BK6QE81.xml");

            StreamResult console = new StreamResult(System.out);
            StreamResult file = new StreamResult(myFile);

            transf.transform(source, console);
            transf.transform(source, file);

        }
            //createIngredients

        private static Node createIngredients(Document doc, String
ingredient_id, String name, String type, String purchasePrice,
                    String stockQuantity) {

            Element ing = doc.createElement("ingredients");

            ing.setAttribute("ingredient_id", ingredient_id);
            ing.appendChild(createElement(doc, "name", name));
            ing.appendChild(createElement(doc, "type", type));
            ing.appendChild(createElement(doc, "purchasePrice",
purchasePrice));
            ing.appendChild(createElement(doc, "stockQuantity",
stockQuantity));

            return ing;
```

```
    }
        //createProduct

    private static Node createProduct(Document doc, String product_id,
String name,
            String price, String[] type, String description) {

        Element pr = doc.createElement("product");

        pr.setAttribute("product_id", product_id);
        pr.appendChild(createElement(doc, "name", name));
        pr.appendChild(createElement(doc, "price", price));

        Node[] node = appendArray(doc, "type", type);

        for (int i = 0; i < type.length; i++) {
            pr.appendChild(node[i]);
        }

        pr.appendChild(createElement(doc, "description", description));

        return pr;

    }
        //createMadeOf
    private static Node createMadeOf(Document doc, String ingredient_id,
String product_id) {

        Element mo = doc.createElement("made_of");
```

```
            mo.setAttribute("ingredient_id", ingredient_id);
            mo.setAttribute("product_id", product_id);

            return mo;

        }
            //createOrder

        private static Node createOrder(Document doc, String order_id, String
date,
        String price, String[] item, String status) {

            Element or = doc.createElement("order");

            or.setAttribute("order_id", order_id);
        or.appendChild(createElement(doc, "date", date));
            or.appendChild(createElement(doc, "price", price));

            Node[] node = appendArray(doc, "item", item);

            for (int i = 0; i < item.length; i++) {
                    or.appendChild(node[i]);
            }

            or.appendChild(createElement(doc, "status", status));

            return or;
```

```
	}
		//createOrderedItems

	private static Node createOrderedItems(Document doc, String order_id,
String product_id, String orderingDate) {

		Element oi = doc.createElement("ordered_items");

		oi.setAttribute("order_id", order_id);
		oi.setAttribute("product_id", product_id);
		oi.appendChild(createElement(doc, "orderingDate",
orderingDate));

		return oi;

	}
		//createCustomer

	private static Node createCustomer(Document doc, String customer_id,
String create_order, String name,
			String postalcode, String street, String houseNumber, String
doorNumber, String phoneNumber, String email,
			String regularCustomer) {

		Element cElement = doc.createElement("customer");

		cElement.setAttribute("customer_id", customer_id);
		cElement.setAttribute("create_order", create_order);
		cElement.appendChild(createElement(doc, "name", name));
```

```
		Element cim = doc.createElement("address");
		cim.appendChild(createElement(doc, "postalcode", postalcode));
	cim.appendChild(createElement(doc, "street", street));
		cim.appendChild(createElement(doc, "houseNumber",
houseNumber));
		cim.appendChild(createElement(doc, "doorNumber",
doorNumber));

	cElement.appendChild(cim);

		cElement.appendChild(createElement(doc, "phoneNumber",
phoneNumber));
		cElement.appendChild(createElement(doc, "email", email));
		cElement.appendChild(createElement(doc, "regularCustomer",
regularCustomer));

		return cElement;

	}
		//createCourier

	 private static Node createCourier(Document doc, String courier_id,
String courier_demand,
	String name, String phoneNumber, String transportType, String
isActive) {

		Element c = doc.createElement("courier");
```

```
            c.setAttribute("courier_id", courier_id);
            c.setAttribute("courier_demand", courier_demand);
            c.appendChild(createElement(doc, "name", name));
            c.appendChild(createElement(doc, "phoneNumber",
phoneNumber));
            c.appendChild(createElement(doc, "transportType",
transportType));
            c.appendChild(createElement(doc, "isActive", isActive));

            return c;

        }
            //createElement

        private static Node createElement(Document doc, String name, String
value) {

            Element node = doc.createElement(name);
            node.appendChild(doc.createTextNode(value));

            return node;

        }

        private static Node[] appendArray(Document doc, String name, String[]
value) {

            Element nodes[] = new Element[value.length];
```

```
            for (int i = 0; i < value.length; i++) {

                nodes[i] = doc.createElement(name);
                nodes[i].appendChild(doc.createTextNode(value[i]));

            }

            return nodes;

        }

}
```